TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Sackler Faculty of Exact Sciences,
Blavatnik School of Computer Science

# Models for Structural and Numerical Alterations in Cancer

THESIS SUBMITTED FOR THE DEGREE OF
“DOCTOR OF PHILOSOPHY”
by

**Ron Zeira**

The work on this thesis has been carried out
under the supervision of
**Prof. Ron Shamir**

Submitted to the Senate of Tel-Aviv University
August 2018

# Acknowledgments

This dissertation summarizes most of my research in the last five years. I would like to express my sincere thanks to my advisor, Ron Shamir, for his guidance, advice and support.

I would also like to thank all the sources who funded my research: Israeli Science Foundation, Dotan Hemato-Oncology Research Center at Tel Aviv University, Bella Walter Memorial Fund of the Israel Cancer Association and Len Blavatnik and the Blavatnik Family foundation. In addition I would like to thank the generous fellowships I received from the Edmond J. Safra Center for Bioinformatics at Tel Aviv University and from the Israeli Center of Research Excellence (I-CORE) Gene Regulation in Complex Human Disease.

I would like to thank all my friends, collaborators and computational genomics lab members for their support and cooperation. Last but definitely not least, I would like to thank my dear family for their support over these years.

# Preface

This thesis is based on the following three articles that were published throughout the PhD period in scientific journals:

1. **Sorting by cuts, joins and whole chromosome duplications**
   Ron Zeira and Ron Shamir
   Published in *Proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching (CPM 2015)* [124] and as a full version in *Journal of Computational Biology (JCB)* [125]

2. **A linear-time algorithm for the copy number transformation problem**
   Ron Zeira, Meirav Zehavi and Ron Shamir
   Published in *Proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)* [95] and as a full version in *Journal of Computational Biology (JCB)* [127]

3. **Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions**
   Ron Zeira and Ron Shamir
   Published in *Bioinformatics* [126]

In addition, the introduction (Sections 1.1-1.4) is based on the review article "**Genome Rearrangement Problems with Single and Multiple Gene Copies: A Review**" by Ron Zeira and Ron Shamir
The article was peer reviewed and will be published as a chapter in the book "Festschrift in honor of Bernard M.E. Moret" (T. Warnow, editor), Springer (2019).

# Abstract

Genome rearrangement problems arise in both species evolution and cancer research. Basic genome rearrangement models assume that the genome contains a single copy of each gene and the only changes in the genome are structural, i.e. reordering of segments. In contrast, numerical changes such as deletions and duplications, which change the number of copies of genes, have been observed in species evolution and prominently in tumorigenesis. In this thesis we describe our studies in which we developed models for structural and numerical alterations in cancer. The models differ in the assumptions taken on the genome structure and in the type of rearrangements allowed during their evolution. We give efficient algorithms and hardness results on these models, and use them to analyze tumor genomes. This thesis advances the state of the art of multi copy genome rearrangements in cancer in two ways. Our models allow a broader set of operations than extant models, thus being more realistic. Furthermore, they attempt to reconstruct the full sequence of structural and numerical events during cancer evolution.

# Contents

# Chapter 1

# Introduction

The computational study of genome rearrangements is a sub-area of computational biology born about 25 years ago [89, 91]. Over that period, it has flourished and developed into a fascinating research area, combining beautiful combinatorial models, elegant theory and applications. Models of the first generation, motivated by species evolution, were simple (though their analysis was sometimes quite sophisticated) and assumed that genomes contain only one copy of each gene. With the explosion of biological data, new analysis opportunities arose, necessitating more complex models and theory.

This chapter describes some of the problems and results related to rearrangement models allowing multiple gene copies. This research area is motivated by evolution of species and of cancer genomes. Studies of cancer genome evolution are stimulated by the recent large scale deep sequencing of thousands of tumor genomes, which has brought about a plethora of novel challenges. Our main focus is on multi-copy models, but key single-copy models are also reviewed briefly for context.

This chapter is by no means exhaustive. The field of modeling genome rearrangements is vast and cannot be covered in one paper. The selection of topics reflects our knowledge (or lack thereof) and taste, and we apologize to the many researchers whose work is not mentioned. For further reading see, e.g., [130, 42, 39].

In the following section we give biological introduction and motivation to genome rearrangements (*GR*) in both species evolution and cancer.[1] Section 1.2 gives computational background and some fundamental results in the analysis of single copy

[1]See the Acronyms chapter for a list of abbreviations

genomes. In Sections 1.3 and 1.4 we review GR models that handle genomes with multiple gene copies in the context of species and cancer evolution.

## 1.1 Introduction to genome rearrangements

### 1.1.1 Genomes and rearrangements

The *genome*[2] encodes instructions used in the development and functioning of all living organisms (bacteria, plants, animals etc.). Genomes are built of *DNA*, a double-stranded molecule in which each *strand* is a long sequence of *nucleotides* (or *bases*). Each base can be of four types *A*, *C*, *G* and *T*. The two strands are *complementary* such that an *A* on one strand is coupled with a *T* on the other strand, and similarly *C* is coupled with *G*. Because of this complementarity, one strand completely determines the other, and DNA molecules are usually represented by the sequence of one strand.

The *genome* is the total DNA material in the cell. It is partitioned into physically disjoint subsequences called *chromosomes*. Chromosomes can be either *linear* and contain two ends called *telomeres*, or *circular*. A *gene* is a segment along the chromosome containing information for the construction of a *protein*. Proteins are molecules that form the "machines" and building blocks of most cellular functions. The direction in which a gene is transcribed into a protein on a given strand determines its *orientation*. Genes are a basic unit of heredity passed from one generation to the other.

The causes of diversity of organisms are changes in the DNA between generations. Such changes, which arise due to inaccurate replication and also due to environmental effects on the DNA, open the possibilities for modified genes, new genes, and eventually new species.

Genomes can evolve in a local and global manner. *Local* alterations refer to *point mutations* in the DNA sequence that can either *substitute* a single base (or a very short subsequence) with a different one, *insert* a single base into the sequence or *delete* a base from the sequence. Such local alterations can also involve very short

[2]Since this chapter concentrates mainly on the computational aspects of GR, we only give a brief biological introduction. We italicize terms that actually require definitions. For concise biological definitions see, e.g., [66]. Box 1 defines some biological terms that are mentioned in the text.

sequence segments. On the other hand, a sequence can also evolve by modifying its organization on a large scale. These *global* mutations, called *genome rearrangements* or *structural variations*, relocate, duplicate, or delete large fragments of the DNA. The main rearrangement types include the following (compare Figure 1.1):

- *Deletion.* A segment of DNA is lost. A *chromosome deletion* is a deletion of an entire chromosome.
- *Inversion* or *reversal.* A segment is cut and reinserted in the opposite orientation. Since the insertion reverses the two strands, the result is an inverted and reverse complemented DNA sequence.
- *Transposition.* A DNA segment is moved to a different location.
- *Duplication.* A genomic segment is copied and reinserted into the genome. In a *tandem duplication* the copy is inserted right after the original one. An arbitrary (non tandem) duplication inserts the new copy at an arbitrary position (one particular type of such is *retrotransposition*). A *whole chromosome duplication* makes another copy of an entire chromosome. A *whole genome duplication* duplicates all the genome's chromosomes.
- *Translocation.* Two linear chromosomes exchange their end segments.
- *Fusion.* Two chromosomes are joined into one.
- *Fission.* A chromosome splits into two chromosomes.

The above rearrangement operations affect DNA segments rather than nucleotides and thus genomes are often represented by sequences of segments in this context. Two segments are called *homologous* if they derive from a common ancestor either by speciation (in that case the segments appear in the genomes of different species) or by duplication (where they occur on the same genome).

While these operations represent the common rearrangements observed in genomes they do not necessarily correspond to atomic biological events. For instance, even though fusions and fissions are observed in genomic data, they may be a result of multiple operations and not a single one.

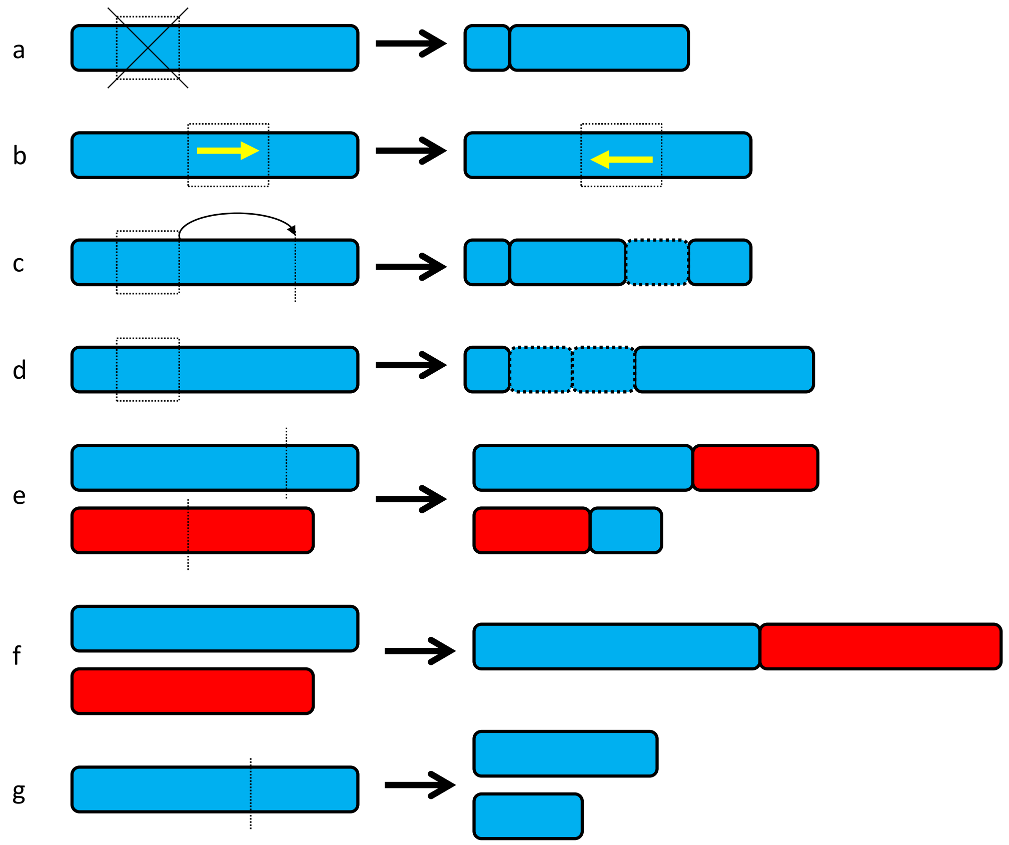

Figure 1.1: *Genome rearrangements.* a. Deletion. b. Reversal. c. Transposition. d. Tandem duplication. e. Translocation. f. Fusion. g. Fission.

### 1.1.2 Genome rearrangements in species evolution

The genomes of related species are very similar. For instance, most of the mouse and human genomes can be divided into segments in which gene content is conserved [26]. However, the order of these segments along the human and mouse genomes is different. This difference is attributed to rearrangement events occurring after the divergence of the two lineages.

The phenomenon of GR in evolution was discovered by Sturtevant and Dobzhansky who demonstrated inversions between genomes of drosophila species [103]. Palmer and colleagues observed that mitochondrial DNA of related plant species have similar gene content but different segment ordering (Figure 1.2) [79, 102]. This immediately raises the question of how this change came about, the fundamental problem that underlies the GR field.

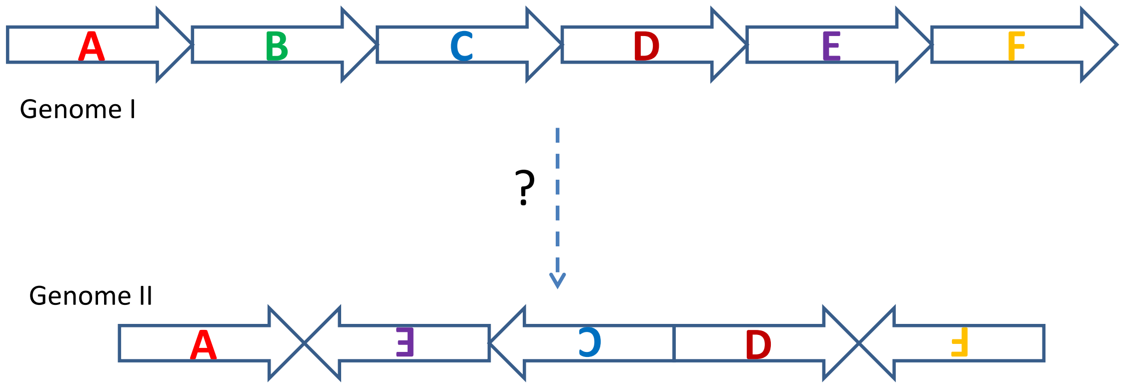

Figure 1.2: The basic sorting problem. Given Genome I and Genome II, and a set of allowed operations, we wish to find a shortest sequence of operations transforming Genome I into Genome II. The sequence is called a sorting scenario and the number of operations in it is called the sorting distance. See Figure 1.3 for a sorting scenario.

The detection of GRs in the studies mentioned was largely based on molecular cytogenetics techniques such as *chromosome banding* and *in-situ hybridization* [84]. These studies mostly focused on relatively close species and a small number of rearrangements between them [92]. With the advent of sequencing technologies, bioinformatic methods enabled locating homologous segments in different genome sequences, thus creating finer comparative maps based on genome sequences [81]. See Box 2 for details on the technologies for rearrangement detection. Note that these techniques do not give evidence to atomic rearrangement events but only measure the final genome.

Sankoff pioneered the computational study of GR in species evolution [89, 91]. The basic assumption of most mathematical models is that evolution is parsimonious and prefers a shortest or most likely sequence of events. In their seminal works, Hannenhalli and Pevzner gave the first polynomial algorithm for the problem of transforming one genome into the other by the minimum number of reversals and of reversals and translocations, respectively [49, 50]. They used their algorithm to give a shortest event sequence between men and mice, and between cabbage and turnip.

Classical computational rearrangement models assume that each gene in the two genomes under study appears only once and that 1-1 homology between the genes of the genomes has been established. While this assumption may hold for closely related genomes, it is unwarranted for divergent species with several copies of the same genes or highly similar genes. Duplications are an important source of new

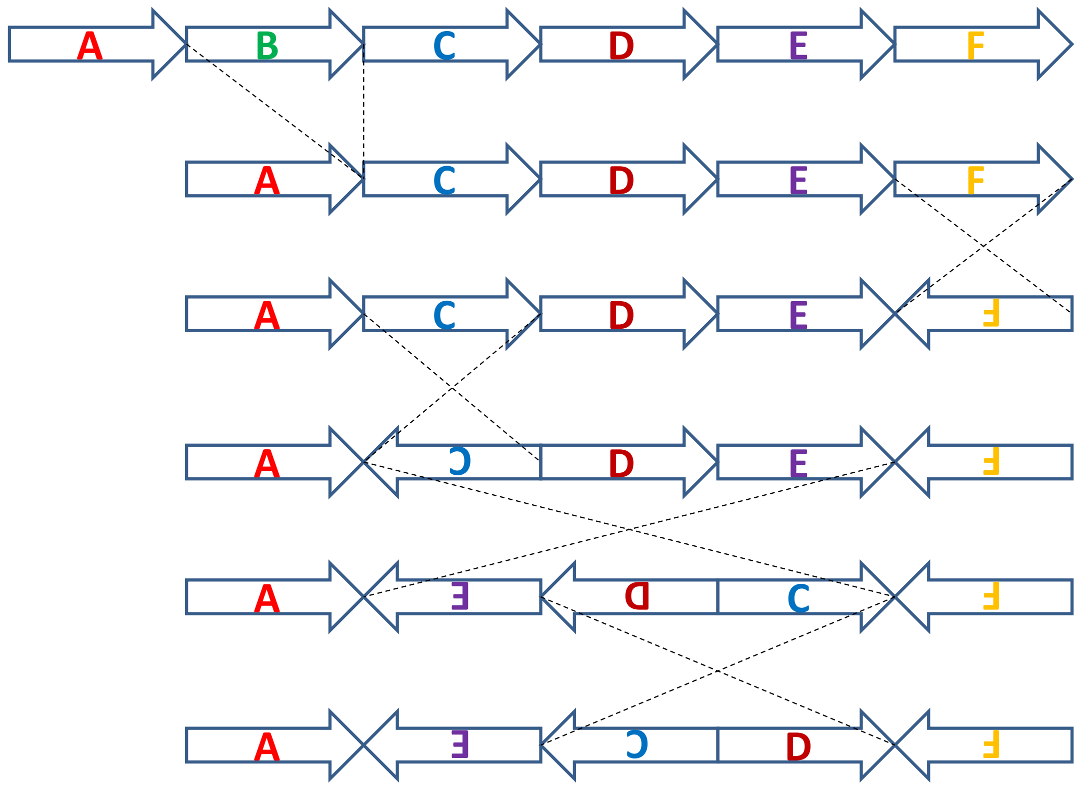

Figure 1.3: A sorting scenario for the chloroplast genome evolution between two conifers. The genome at the top is transformed to the one at the bottom in five steps. The first is a deletion and the next four are inversions of genomic segments. The ends of the involved segments are indicated by the broken lines. Adapted and simplified from Strauss *et al.* [102].

gene functions since new gene copies tend to diverge through mutations and develop new functions. For instance, evidence of whole genome duplication events have been observed in most angiosperm genomes [20].

### 1.1.3 Genome rearrangements in cancer

Cancer is a complex disease driven by the accumulation of somatic DNA mutations over generations of cell divisions. Such mutations affect tumor growth, clinical progression, immune escape, and drug resistance [31].

Mutations in cancer cells can be local, affecting single DNA base pairs. These

**Box 1 Some biological jargon**

| | |
|---|---|
| Angiosperms | the flowering plants |
| Chloroplasts | specialized compartments in plant cells responsible for photosynthesis |
| Conifers | cone-bearing seed plants |
| Drosophila | fruit fly |
| Metaphase | a stage in cell division. During metaphase chromosomes can be distinguished under the microscope after appropriate painting |
| Orthologs | descendant copies of the same gene sequence in different species. Orthologs can usually be identified by their sequence similarity |
| Somatic cell | any cell forming the organism body other than the reproductive cells. The genome in sperm and egg cells is inherited in sexual reproduction, along with any mutations in it. In contrast, the genome of somatic cells is not inherited, but mutations in cancer genomes are inherited in cell division. |
| Somatic mutation | a mutation occurring in somatic cells. |

mutations, called *single nucleotide variants* (*SNV*), can number in the thousands per cancer cell. On the other hand, large scale mutations, i.e. GRs, can relocate fragments of the DNA. Aberrations that change the amount of genomic content, called *copy number alterations* (*CNAs*) include duplications and deletions of genomic regions. The *karyotype* of a cell is its complete set of chromosomes, consisting of the number and structure of the chromosomes in it. Large-scale aberrations can have a dramatic effect on the cancer karyotype (see Figure 1.4).

Somatic mutations may amplify genes that promote cancer (*oncogenes*) or harm genes that inhibit cancer development (*tumor suppressor* genes). In addition, rearrangements such as translocations and inversions may change gene structure and regulation and create novel fusion genes, with or without additional changes in copy number (CN).

Cancer is an evolutionary process in which a normal genome accumulates mutations that eventually transform it into a cancerous one [11]. The gain of advantageous mutations leads to a *clonal expansion*, forming a larger population of the mutated cells. Subsequent clonal expansions occur as additional advantageous mutations accumulate in descendant cells. A single tumor biopsy will often contain a mixture of several competing tumor clones. These tumor clones frequently differ in

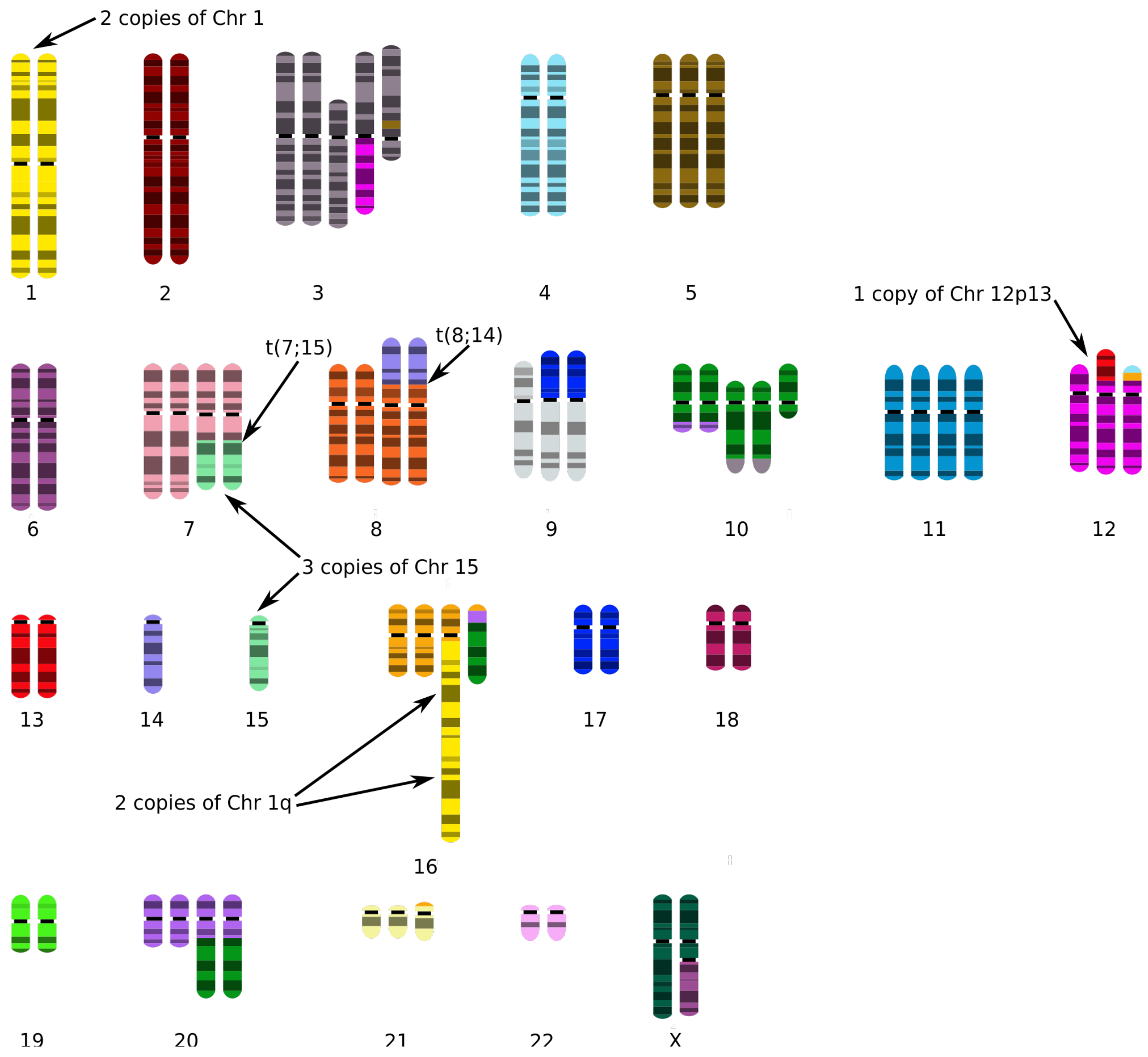

Figure 1.4: A schematic of the karyotype of the T47D breast cancer cell line. The chromosome numbers in the normal diploid are indicated below each subfigure. In a normal karyotype, each chromosome has two copies, as for Chr. 4, 13, 17 and 18. Among the GRs in this cancer genome we see chromosomal duplications (e.g., four copies of Chr. 11), translocations (between Chr. 8 and Chr. 14), and more complex events (e.g., tandem duplication of one arm of Chr. 1 and fusion with an extra arm of Chr. 16). Image source: [10] and Wikimedia Commons [52]. This image is used under license CC BY-SA 3.0.

their genomic content and structure. When sequencing the tumor, one actually obtains a mixture of several tumor clones and of normal cells. Recent research suggests that this heterogeneity has profound clinical implications [31].

**Box 2 Detection of genome rearrangements**

The classical ways to detect chromosomal abnormalities in cytogenetics are *G-banding* and *fluorescence in situ hybridization* (*FISH*), which allow viewing the chromosome in metaphase at low resolution [83]. FISH measures the CN of tens to hundreds of targeted genes [28]. *Array comparative genomic hybridization* (*array CGH*) gives a higher resolution of CN estimation for a cell population [110].
Today, next generation sequencing techniques are the main data source for cancer mutation analyses [32]. Whole genome sequencing provides tens to hundreds of millions of DNA reads that enable the detection of variants. These short reads are assembled into longer DNA sequences and alignment to a reference genome can determine sequence similarity and structural changes. This reference genome can be of a related species for evolutionary studies or of a normal tissue in the case of cancer.
Paired-end read technologies generate pairs of short reads such that the approximate distance between them and their relative orientations in the target genome are known. Read pairs in which the location or orientation in the reference genome is not as expected are called *discordant*. These reads give evidence of structural rearrangement operations [75]. The *read depth* data, i.e. the number of concordant reads mapped to each region in the reference genome, can also be used to assess CN and CNAs [75].

## 1.2 Single gene models, operation types and distance measures

In this section we give a brief introduction to GR models. We start by giving the definitions and terminology used in computational GR analysis. We then review several classical single gene models.

### 1.2.1 Genome representation

Here we describe simple mathematical representations of genomes for GR analysis. A genome representation should preserve the information about the order, orientation and homology between segments (see Figure 1.3). In some representations, different copies of similar segments can be distinguished while in other representations they cannot. For instance, the two copies of chromosome 1 in Figure 1.4 are indistinguishable. On the other hand, in some cases gene copies can be distinguished from one another, for example due to gene sequence changes since its speciation. Different GR models may use different representations depending on the model assumptions or data used.

Consider a set $\mathcal{G}$ of $n$ segments in the genome. For convenience, we call the segments in $\mathcal{G}$ *genes*, though they do not necessarily represent biological gene entities , i.e., sequence segments that encode proteins. A gene $g$ is an oriented sequence of DNA that starts with a *tail* and ends with a *head*, denoted as $g_t$ and $g_h$ respectively. The default orientation of a gene, and thus its head and tail, can be determined arbitrarily or according to its transcription in some reference genome. The set of *extremities* of the genes is $\mathcal{E} = \{g_t | g \in \mathcal{G}\} \cup \{g_h | g \in \mathcal{G}\}$.

An *adjacency* between two consecutive genes in a genome is an unordered pair of extremities. Thus, an adjacency between two genes $a, b \in \mathcal{G}$ can take one of four forms, depending on their orientation: $\{a_h, b_t\}, \{a_t, b_h\}, \{a_t, b_t\}, \{a_h, b_h\}$. An extremity that is not adjacent to another extremity is called a *telomere*, and is represented by a singleton set, e.g. $\{a_h\}$.

In some formulations, a gene may have multiple copies corresponding, for example, to homologous yet distinguishable genes. The copies of such a gene $g \in \mathcal{G}$ are identified by a superscript. For example, $g^1, g^2, g^3$ are three distinct copies of gene $g$. Such a gene with multiple distinguishable copies is called a *labeled* gene. A gene that has a single copy or has multiple indistinguishable copies is called *unlabeled*. For a gene $g$, we call the number of copies it has its *copy number* and denote it by $cn(g)$. A gene set $\mathcal{G}$ with one copy for each gene is called an *ordinary gene set*. A *labeled gene set* is a set $\mathcal{G}^L = \{g^i | g \in \mathcal{G}, 1 \leq i \leq cn(g)\}\}$ and an *unlabeled gene set* $\mathcal{G}^U$ is a multiset $\mathcal{G}^U = \cup_{g \in \mathcal{G}} \cup_{1 \leq i \leq cn(g)} \{g\}$. For instance, $\mathcal{G}^L = \{a^1, a^2, b^1, c^1, c^2, c^3\}$ and $\mathcal{G}^U = \{a, a, b, c, c, c\}$ are labeled and unlabeled gene sets, respectively, that have two copies of gene $a$, one of $b$ and three of $c$. Similar to genes, extremities belonging to labeled genes are distinguishable (e.g., $a_h^1 \neq a_h^2$), while extremities of unlabeled gene are indistinguishable. Furthermore, unlabeled heads and tails of the same unlabeled gene cannot be matched. In other words, we do not know which tail and head come from the same gene copy.

A *labeled genome* $\Pi$ over a labeled gene set $\mathcal{G}^L$ is a set of adjacencies and telomeres such that every labeled extremity $e^i \in \mathcal{E}^L$ appears exactly once in an adjacency or telomere of $\Pi$. Similarly, an *unlabeled genome* $\Pi$ over an unlabeled gene set $\mathcal{G}^U$ is a multiset of adjacencies and telomeres such that every unlabeled extremity $e \in \mathcal{E}^U$ of gene $g$ appears exactly $cn(g)$ times in adjacencies or telomeres of $\Pi$.

$\Pi = \{\{a_t^1\}, \{a_h^1, b_h^1\}, \{b_t^1, c_t^1\}, \{c_h^1\}, \{a_t^2\}, \{a_h^2, b_t^2\}, \{b_h^2\}, \{b_t^3\}, \{b_h^3, c_h^2\}, \{c_t^2\}\}$ and $\Gamma = \{\{a_t\}, \{a_h, b_h\}, \{b_t, c_t\}, \{c_h\}, \{a_t\}, \{a_h, b_t\}, \{b_h\}, \{b_t\}, \{b_h, c_h\}, \{c_t\}\}$ are examples of labeled and unlabeled genomes. If the gene set of a genome is ordinary, we call it an *ordinary genome* (or a *single copy genome*).

The *graph representation* of a genome $\Pi$ is an undirected graph $G_\Pi = (\mathcal{E}, E)$. Its nodes are the extremities of $\Pi$ (either labeled or unlabeled) and $E$ consists of interval edges and adjacency edges. An *interval edge* connects the head and tail of a gene. For unlabeled genomes there are $cn(g)$ parallel interval edges of the edge $(g_h, g_t)$ for every gene $g$. For labeled genomes, each labeled gene copy $g^i$ has a single interval edge $(g_h^i, g_t^i)$. *Adjacency edges* connect the extremities $x$ and $y$ where $\{x, y\}$ is an adjacency of $\Pi$. We call $G_\Pi$ the *genome graph* of $\Pi$. The representations $\Pi$ and $G_\Pi$ are equivalent and thus we use them interchangeably. Notice that for each node (extremity), its number of interval edges (*interval-degree*) equals its number of adjacency edges (*adjacency-degree*) plus the number of telomeres it belongs to. Figures 1.5, 1.6 and 1.7 show genome graphs for ordinary, labeled and unlabeled genomes, respectively.

An *alternating route* in $G_\Pi$ is either a path or a cycle in which no two consecutive edges are of the same type (interval/adjacency). A *chromosome decomposition* $D_\Pi$ of the genome $\Pi$ is a decomposition of $G_\Pi$ into a set of edge disjoint maximal alternating cycles and alternating paths that cover all edges. Note that a chromosome decomposition is always possible since the interval-degree is equal to the adjacency-degree for every node that is not in a telomere, and that maximal paths must start and end with telomeres. Labeled and ordinary genomes have a unique chromosome decomposition by simply taking the set of connected components, since the interval-degree and the adjacency-degree of every non-telomere node is 1 (see Figures 1.5 and 1.6). There may be several decompositions for a multi-copy unlabeled genome (see Figure 1.7). Each alternating route in a decomposition is called a *chromosome*. A chromosome is called *circular* if the corresponding route is a cycle, and *linear* otherwise. A decomposition is called *linear* if all its chromosomes are linear, *circular* if all its chromosomes are circular, and otherwise *mixed*. Figure 1.5 shows an ordinary genome with one linear and one circular chromosome. An ordinary genome composed of a single linear chromosome is called a *signed permutation*.

A *signed genomic string* is a sequence of oriented genes, e.g. $1\ -2\ 3$. For a chromosome $C \in D_\Pi$, we define the *chromosome string* of $C$ as follows. Start at one of the ends of a linear chromosome with the string '('. Traverse the route until all edges along the route are covered. For each traversal of an interval edge from a tail $g_t$ to a head $g_h$ append $g$ to the string. For traversal from $g_h$ to $g_t$ append $-g$ to the string. After finishing the traversal, append the string with ')'. For a chromosome string $C$, let $-C$ be the chromosome string in which the order and orientation of all gene are in inverted, e.g. if $C = (1\ 2\ 3)$ then $-C = (-3\ -2\ -1)$. $C$ and $-C$ are *equivalent* as they correspond to the same set of adjacencies. For a circular chromosome, do the same starting from an arbitrary extremity interval edge without appending brackets. The resulting sequence is cyclic and all shifts and inversions of it are equivalent. We use $<>$ to denote circular genomes (Figures 1.5, 1.6 and 1.7).

A *string representation of a genome decomposition* $D_\Pi$ is the multi-set of chromosome strings for each chromosome in the decomposition (Figures 1.5, 1.6 and 1.7). Two string representations are equivalent if there is a bijective mapping between equivalent chromosome strings in them. For labeled and ordinary genomes, the string representation is unique (up to equivalence) and therefore we sometime use this representation.

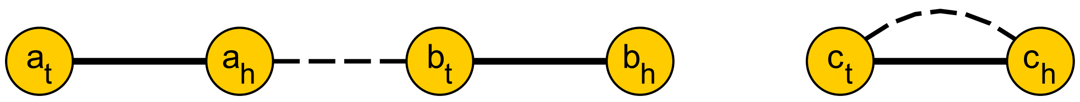

Figure 1.5: A genome graph $G_\Pi$ of an ordinary genome $\Pi = \big\{\{a_t\}, \{a_h, b_t\}, \{b_h\}, \{c_t, c_h\}\big\}$. Bold edges correspond to interval edges; dashed edges correspond to adjacencies. Since $\Pi$ is an ordinary genome, it has a unique decomposition $D_\Pi$ whose string representation is $\big\{(a\ b), < c >\big\}$

Given an unlabeled genome $\Pi$ over the gene set $\mathcal{G}^U$, a *labeling* of $\Pi$ produces a labeled genome $\Gamma$ over the gene set $\mathcal{G}^L$ such that distinct gene copies of a gene $g$ are mapped to distinct labeled genes $g^1, \ldots, g^{cn(g)}$ in $\mathcal{G}^L$. For example, the labeled genomes in Figure 1.6 and 1.7B are two possible labelings of the unlabeled genome in Figure 1.7A. We denote $L(\Pi)$ to be the set of all possible labelings of $\Pi$.

Given a genome $\Pi_1$ over the gene set $\mathcal{G}_1$, an *operation* creates a new genome $\Pi_2 \neq \Pi_1$ over a new gene set $\mathcal{G}_2$. An operation is said to be *structural* if $\mathcal{G}_1 = \mathcal{G}_2$. An operation is said to be *numerical* if the CN of some gene is different under $\mathcal{G}_1$ and $\mathcal{G}_2$. Notice that a structural operation only changes the structure, i.e. $\Pi_2 \neq \Pi_1$,

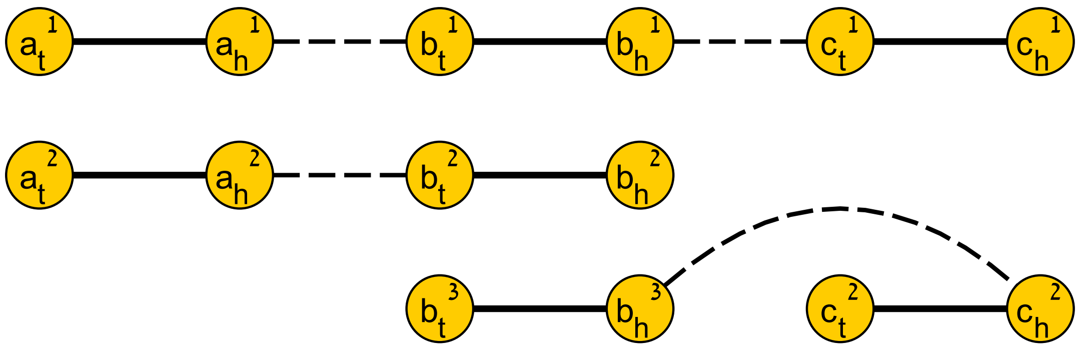

Figure 1.6: A genome graph $G_\Delta$ of a labeled genome $\Delta = \{\{a_t^1\},\{a_h^1,b_t^1\},\{b_h^1,c_t^1\},\{c_h^1\},\{a_t^2\},\{a_h^2,b_t^2\},\{b_h^2\},\{b_t^3\},\{b_h^3,c_h^2\},\{c_t^2\}\}$. Bold edges correspond to interval edges; dashed edges correspond to adjacencies. Since $\Delta$ is an ordinary genome, it has a unique decomposition $D_\Delta$ whose string representation is $\big\{(a^1\ b^1\ c^1),(a^2\ b^2),(b^3\ {-c^2})\big\}$

whereas a numerical operation also changes the gene set, i.e. $\mathcal{G}_1 \neq \mathcal{G}_2$.

A *genome rearrangement model* is composed of a set of allowed operations $\mathcal{O}$ and additional constraints on genomes. A *sorting scenario* of length $d$ from $\Pi$ into $\Gamma$ is a series of genomes $\Pi_0,\ldots,\Pi_d$ such that $\Pi_0 = \Pi, \Pi_d = \Gamma$ and for each $i$, $\Pi_{i+1}$ is a legal genome (under the model constraints) that is a result of an allowed operation on $\Pi_i$. The *sorting distance* is the length of a shortest sorting scenario from $\Pi$ into $\Gamma$. We call $\Pi$ the *source* genome and $\Gamma$ the *target* genome. The *sorting problem* under model $\mathcal{O}$ receives as input $\Pi$ and $\Gamma$, and looks for a sorting scenario of minimum length from $\Pi$ to $\Gamma$. Figure 1.3 shows a sorting scenario of length 5 from $(A\ B\ C\ D\ E\ F)$ to $(A\ {-E}\ {-C}\ D\ {-F})$ in a model allowing deletions and inversions.

## Operations

**Reversal**. An *inversion* of a signed genomic string reverses the string and multiplies all elements by $-1$. Hence the inversion of $(2\ {-3}\ 5\ {-1})$ denoted as $-(2\ {-3}\ 5\ {-1})$, is $(1\ {-5}\ 3\ {-2})$. For a string $S = s_1 \ldots s_n$, $S[i,j]$ is the substring $s_i \ldots s_j$. Let $C$ be a chromosome string. A *reversal* $\rho(i,j)$ inverts $C[i,j]$, resulting in a new chromosome $C' = C[1,\ldots,i-1] \cdot -C[i,\ldots,j] \cdot C[j+1,\ldots,m]$, where $\cdot$ is the concatenation operator. For example, $\rho(3,5)$ of $C = (1\ 3\ 2\ 4\ 5\ 6)$ is $C' = (1\ 3\ {-5}\ {-4}\ {-2}\ 6)$. Reversals can be similarly defined on a single chromosome in the genome graph, by

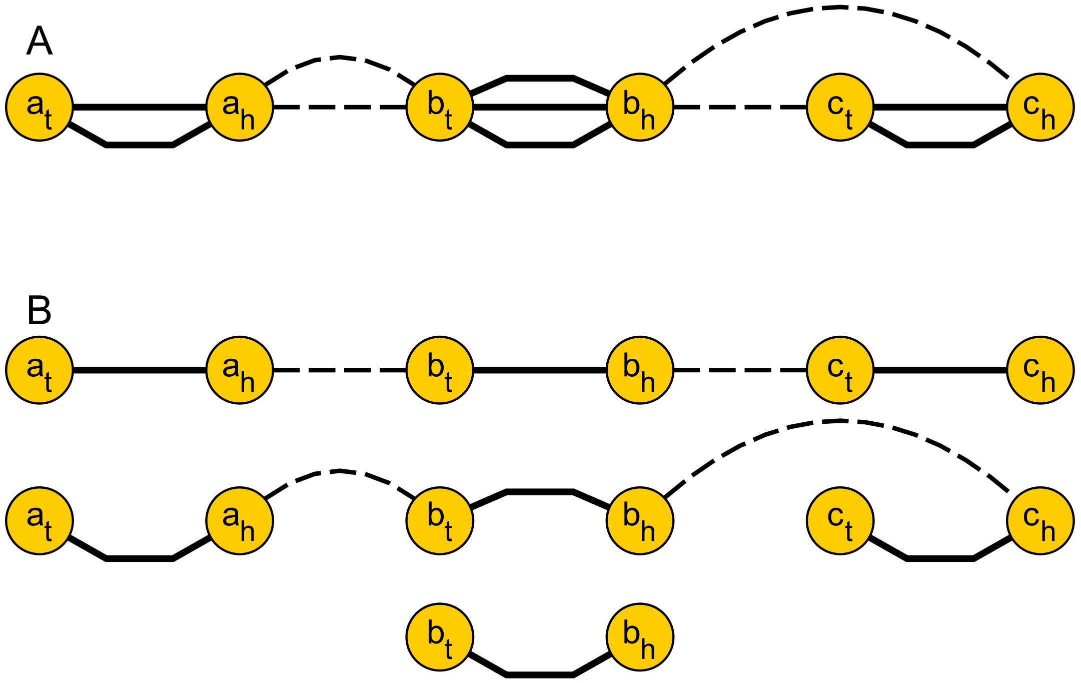

Figure 1.7: **A**: A genome graph $G_\Gamma$ of an unlabeled genome $\Gamma = \{\{a_t\}, \{a_h, b_t\}, \{b_h, c_t\}, \{c_h\}, \{a_t\}, \{a_h, b_t\}, \{b_h\}, \{b_t\}, \{b_h, c_h\}, \{c_t\}\}$. Bold edges correspond to interval edges; dashed edges correspond to adjacencies. **B**: One possible decomposition $D^1_\Gamma$ of $\Gamma$, whose string representation is $\big\{(a\ b\ c), (a\ b\ -c), (b)\big\}$. A different decomposition $D^2_\Gamma$ corresponding to $\big\{(a\ b\ c), (a\ b), (b\ -c)\big\}$ can be seen in Figure 1.6 by suppressing the superscripts.

cutting two adjacencies and reconnecting the loose extremities such that the result is a linear chromosome. A reversal on a labeled or ordinary genome is a reversal on one of its chromosomes. Reversals for general (not ordinary) unlabeled genomes are not defined as they may have several chromosome decompositions. See Figure 1.1B and Figure 1.8A,B.

**Translocation**. Let $C = c_1 \ldots c_m$ and $D = d_1 \ldots d_n$ be two linear chromosomes in string representation of an ordinary or labeled genome. A *translocation* $tr(C, D, i, j)$ transforms $C$ and $D$ into two new chromosomes, either $C[1, \ldots, i] \cdot D[j+1, \ldots, n]$ and $D[1, \ldots, j] \cdot C[i+1, \ldots, m]$, or $C[1, \ldots, i] \cdot -D[1, \ldots, j]$ and $-C[i+1, \ldots, m] \cdot D[j+1, \ldots, n]$. That is, the adjacencies $C_i, C_{i+1}$ and $D_j, D_{j+1}$ are cut, and the four loose ends are reconnected in a new way. An equivalent definition can be made on chromosome graphs, i.e., breaking an adjacency on each chromosome and reconnecting the nodes (see Figure 1.1e and Figure 1.9). Again notice that translocations are not uniquely defined for general unlabeled genomes.

**DCJ**. A *double-cut-and-join* (*DCJ*) is an operation that cuts two adjacencies and reconnects the four loose ends in a new way into two adjacencies. It can be applied on labeled and unlabeled genomes. A DCJ can take one of the following forms:

1. If adjacencies $\{p, q\}, \{r, s\} \in \Pi$ are cut, replace them with either $\{p, r\}, \{q, s\}$ or $\{p, s\}, \{r, q\}$ (Figures 1.8, 1.9).

2. If adjacency $\{p, q\} \in \Pi$ is cut and telomere $\{r\} \in \Pi$ is involved, replace them with either $\{p, r\}, \{q\}$ or $\{r, q\}, \{p\}$ (Figure 1.10).

3. If telomeres $\{q\}, \{r\} \in \Pi$ are involved, replace them with an adjacency $\{r, q\}$ thereby joining the two chromosomes (Figure 1.11). This operation is referred as a *fusion* or a *join*.

4. If adjacency $\{p, q\} \in \Pi$ is cut and an empty adjacency is involved, replace them with two telomeres $\{p\}, \{q\}$ (Figure 1.11). Hence, a linear chromosome containing the adjacency is cut into two chromosomes, or becomes linear if it was circular. This operation is referred as a *fission* or a *cut*.

Note that a DCJ realizes both reversals (when the two adjacencies come from the same chromosome) and translocations (when they are from different chromosomes). When the adjacencies that are cut are from the same chromosome the result of a DCJ can also be splicing out of a segment between the cuts into a separate cyclic chromosome. This circular excision is somehow artificial and does not necessarily correspond to real biological phenomenon.

**SCoJ**. A *single-cut-or-join* (*SCoJ*) operation either cuts an adjacency or joins two telomeres, respectively (Figure 1.11).

In the next section we briefly review basic results on ordinary genome models. As our focus is primarily on multiple-copy problems, we only skim selected results. The interested reader can find much more information on this topic in [130] and [42].

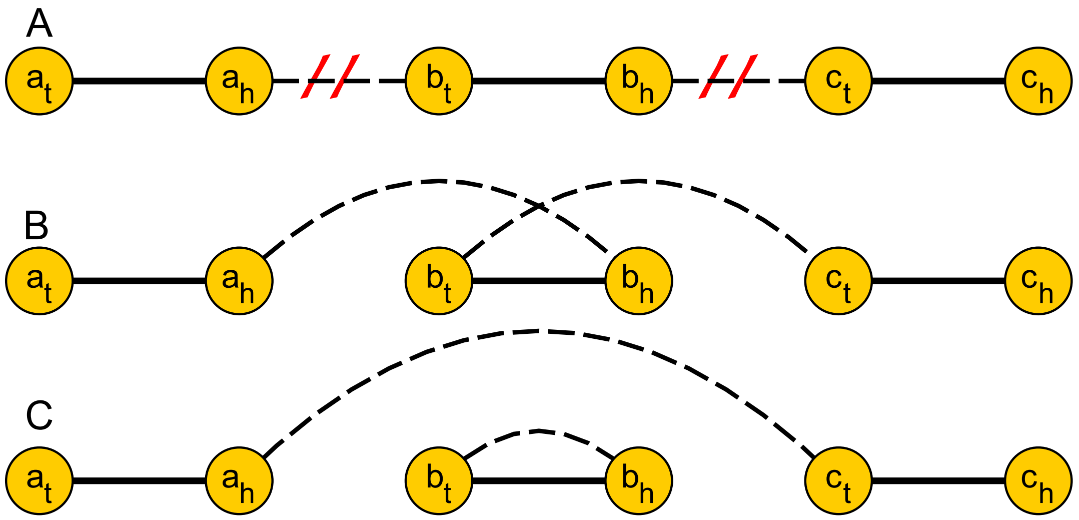

Figure 1.8: Reversal and DCJ. **A**: The genome graph of $(a\ b\ c)$; the two diagonal stripes correspond to the cut adjacencies. **B**: The genome $(a\ -b\ c)$ is a result of a reversal or a DCJ. **C**. The genome $\{(a\ c), < b >\}$ corresponds to the other DCJ option.

### 1.2.2 Breakpoint distance

The *breakpoint (BP) distance* is a simple measure of dissimilarity between two genomes that is not related to a specific type of operation. Generally speaking, the breakpoint distance measures the number of adjacencies and telomeres that are in one genome but not in the other. The breakpoint distance has several definitions depending on the different weights of common adjacencies and telomeres [82, 106].

For two ordinary genomes $\Pi$ and $\Gamma$ over the same $n$ genes, Tannier *et al.* [106] give the following formula for the breakpoint distance:

$$d_{BP} = n - (A + E/2) \tag{1.1}$$

where $A$ is the number of common adjacencies and $E$ the number of common telomeres of $\Pi$ and $\Gamma$. Clearly, the distance is computable in linear time.

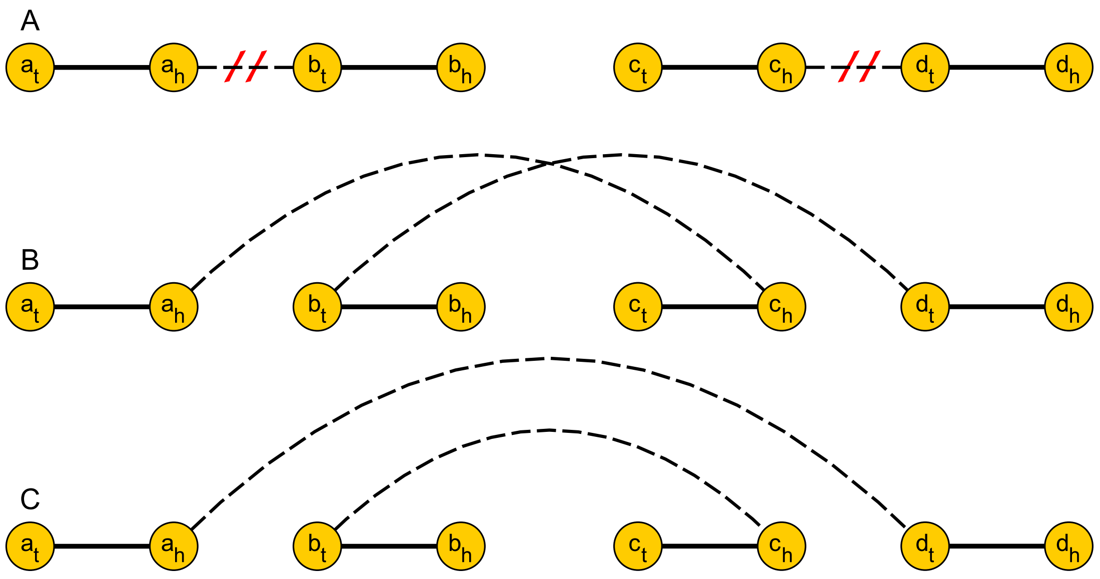

Figure 1.9: Translocation. **A**: Two chromosomes $\{(a\ b), (c\ d)\}$; the two diagonal stripes correspond to the cut adjacencies. **B**,**C**: Two possible translocations (or DCJs) corresponding to $\{(a\ -c), (-b\ d)\}$ (B) and $\{(a\ d), (c\ b)\}$ (C).

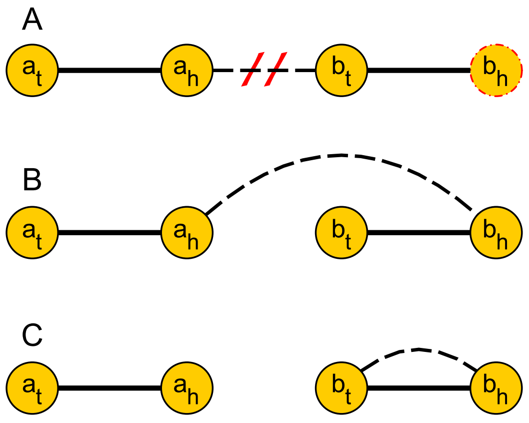

Figure 1.10: DCJs on telomeres. **A**: Chromosome $(a\ b)$; the diagonal stripes and the dotted circle show the cut adjacency and telomere involved. **B**,**C**: Two possible DCJs corresponding to $(a\ -b)$ (B), i.e. reversal, and $\{(a), < b >\}$ (C).

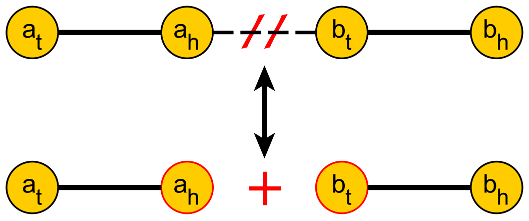

Figure 1.11: Single cut or join. A cut breaks an adjacency into two telomeres corresponding to the transition from the top to the bottom genome. A join is the reverse operation corresponding to the transition from the bottom to the top genome.

### 1.2.3 Reversal and translocation distances

Given signed permutations $\Pi$ and $\Gamma$ over the same $n$ genes, we seek a shortest sequence of reversals from $\Pi$ into $\Gamma$. We can assume w.l.o.g. that $\Gamma$ is the identity permutation $(1 \dots n)$.

Sorting signed permutations by reversals is undoubtedly the most famous GR problem [9]. In their seminal work, Hannenhalli and Pevzner gave the first polynomial time algorithm for the problem [51]. Since then, the theory was greatly simplified [16, 59, 6, 12]. Bader, Moret and Yan have shown that finding the reversal distance can be done in linear time [6], whereas computing a shortest sorting scenario can be done in $O(n^{3/2})$ [46, 105]. Interestingly, sorting *unsigned* permutations (i.e., without gene orientations) by reversals is NP-hard [24].

The problem of sorting multi-chromosomal genomes by translocations was first introduced by Kececioglu and Ravi [61]. Hannenhalli [48] gave the first polynomial time algorithm for the problem and an improved, linear time algorithm was introduced by Bergeron *et al.* [14].

Sorting by reversals and translocations was proved to be polynomial by Hannenhalli and Pevzner [50], who reduced the problem to sorting by reversals. The theory and algorithm were later slightly corrected and revised [108, 76, 77, 55, 15]. The algorithm was used to compute for the first time a sorting scenario and distance between the mouse and human genome [50]. Interestingly, the distance achieved closely matched a prediction by Nadeau and Taylor from the 1980s [71]. Efficient implementations of the algorithms for sorting by reversals and translocations are available as part of the *GRAPPA* [6] and *GRIMM* [109] tools. Those tools also use the ability to compute exact pairwise distances efficiently in order to compute a tree of evolution by reversals and translocations among multiple species, albeit heuristically.

The main representation used for the analysis of this problem (and other rearrangement models) is the Breakpoint Graph ($BG$). Given two genomes $\Pi$ and $\Gamma$, the *breakpoint graph* $BG(\Pi, \Gamma)$ is an undirected graph whose nodes are the extremities of both genomes, and whose edges are the adjacencies of both genomes distinguished by color. Edges corresponding to $\Pi$ ($\Gamma$) adjacencies are called red or $\Pi$-edges (blue or $\Gamma$-edges, respectively). See an example in Figure 1.12.

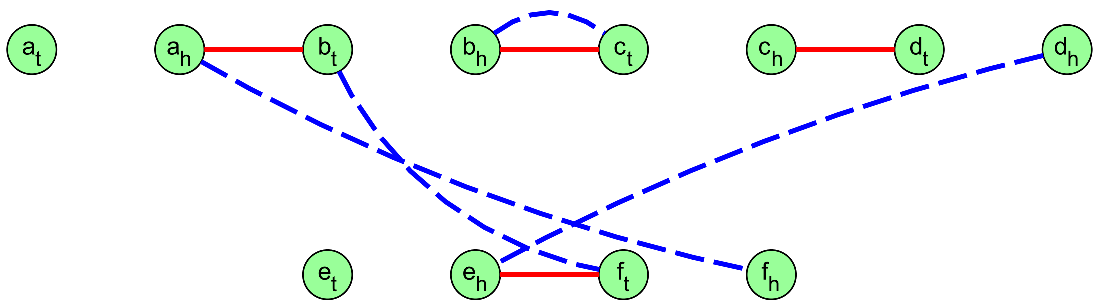

Figure 1.12: A breakpoint graph for $\Pi = \{(a \ \ b \ \ c \ \ d), (e \ \ f)\}$ and $\Gamma = \{(a \ -f \ b \ c), (d \ -e)\}$. $\Pi$-edges are solid; $\Gamma$-edges are dashed;

Hannenhalli and Pevzner [51] gave a formula for the reversal distance between signed permutations based on the number of cycles in the BG and certain structures in it called "hurdles" and "fortresses". The distance formula for sorting by reversals and translocations has been devised over the years and depends on more complex structures in the BG [50, 108, 76, 55, 15]. The definitions of these structures are beyond the scope of this review, so the exact distance formulas are omitted. Bergeron [12], and Jean and Nikolski [55], give fairly elementary presentations for sorting by reversals and sorting by reversals and translocations, respectively, including good expositions of structures and the distance formulas.

### 1.2.4 DCJ distance

The inputs for this model are two ordinary genomes $\Pi$ and $\Gamma$ over the same set of $n$ genes. The operations allowed in this model are DCJs.

The DCJ operation, introduced by Yancopoulos *et al.* [118], has gained much attention in GR models in the last decade. The reason is that DCJs capture both reversals and translocations while allowing much simpler algorithms. On the other hand DCJs also allow splicing out circular sub-chromosomes, and fusions and fissions, which have less evidence as atomic biological events. Both the distance and an optimal sorting scenario can be computed in linear time [13].

In the analysis of this problem a new graph representation was introduced. The *adjacency graph* $AG(\Pi, \Gamma)$ of genomes $\Pi$ and $\Gamma$ is a bipartite undirected multigraph whose set of nodes are the adjacencies and telomeres of $\Pi$ and $\Gamma$. Therefore each node is a set of one or two extremities. Nodes belonging to $\Pi$ ($\Gamma$) are called red- or $\Pi$-nodes (blue- or $\Gamma$-nodes, respectively). For every $\Pi$-node $u$ and $\Gamma$-node $v$, there are $|u \cap v|$ edges between $u$ and $v$, i.e., there is an edge for each common extremity between the two nodes. Note that $BG(\Pi, \Gamma)$ is the line graph of $AG(\Pi, \Gamma)$ and vice versa. (The line graph of $G = (V, E)$ is the graph on $E$ in which $x, y \in E$ are adjacent as vertices iff they are adjacent as edges in $G$). See Figure 1.13.

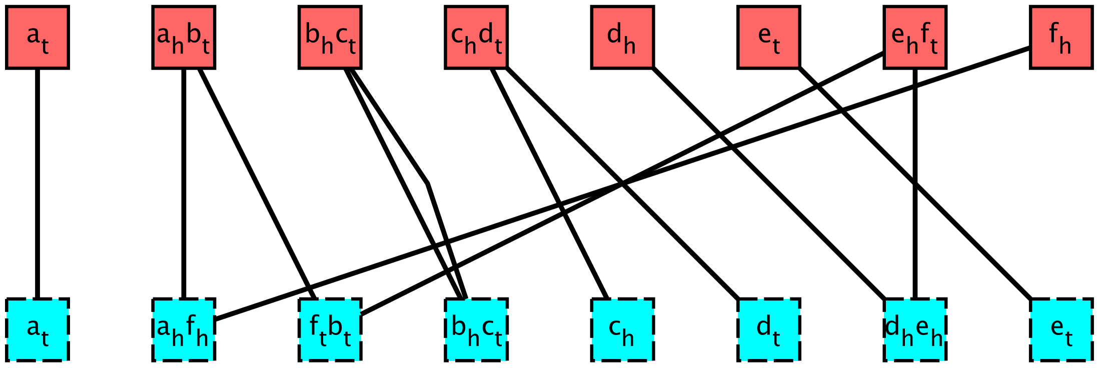

Figure 1.13: An adjacency graph for $\Pi = \{(a\ \ b\ \ c\ \ d), (e\ \ f)\}$ and $\Gamma = \{(a\ {-f}\ b\ c), (d\ {-e})\}$. $\Pi$-nodes are solid; $\Gamma$-nodes are dashed;

Bergeron *et al.* [13] prove that for ordinary genomes $\Pi$ and $\Gamma$ defined over the same set of $n$ genes:

$$d_{DCJ} = n - (C + I/2) \tag{1.2}$$

where $C$ is the number of cycles and $I$ the number of odd length paths starting and ending in telomeres in $AG(\Pi, \Gamma)$. For example, the AG in Figure 1.13 has one cycle and two odd length paths. Thus, since there are 6 genes, the DCJ distance between the two genomes in this case is 4. Notice that there are two additional even length paths in the graph but they do not affect the distance formula.

### 1.2.5 SCoJ distance

The inputs for this model are two ordinary genomes $\Pi$ and $\Gamma$ over the same set of $n$ genes. The operations allowed in this model are SCoJs.

Similar to DCJ, the SCoJ distance and scenario can be found in linear time [41]. Some rearrangements problems for which no polynomial solution is known for DCJ and other operations, are known to be tractable for SCoJ distance. We give examples of such problems in Section 1.3.1. While this simplistic rearrangement distance does not correspond to real biological events, it has been shown to corroborate with the evolutionary distance [17].

For two ordinary genomes $\Pi$ and $\Gamma$ over the same $n$ genes, let $A_\Pi$ ($A_\Gamma$) be the set of adjacencies of $\Pi$ ($\Gamma$, respectively). The SCoJ distance is given by [41]:

$$d_{SCoJ} = |A_\Pi| + |A_\Gamma| - 2|A_\Pi \cap A_\Gamma| \tag{1.3}$$

## 1.3 Multi copy models in evolution

This section discusses multi-copy GR models inspired by species evolution. In Section 1.3.1 we present models allowing whole genome duplication events, but no other copy number changes. The models in Section 1.3.2 allow for the insertion and deletion of new genomic segments but do not account for multiple copies of segments. Models in Section 1.3.3 handle genomes with multiple copies of each gene but do not allow numeric operations. Section 1.3.4 describes a few models that can handle genomes with multiple gene copies and allow numerical operations such as deletions or duplications.

We limit our discussion here to distance problems between two genomes. We refer the reader to the review by El-Mabrouk and Sankoff on the analysis of gene order evolution beyond single-copy genes [39], which discusses in depth the phylogenetic aspects of GR models in the context of species evolution.

### 1.3.1 Polyploidy

We discuss here problems motivated by *whole genome duplication* (*WGD*) events in species evolution. WGD is viewed as a fundamental step in evolution, as doubling of the gene contents allows great diversification of gene functions. For example, strong evidence for WGD events was reported for yeast [117] and for plant genomes [20]. The basic question tackled by these formulations is finding a shortest sorting scenario between a given ancestral genome (before or right after WGD) and a given extant genome under GD models allowing only structural operations. In addition, an underlying assumption is that the extant genome has significantly evolved since the duplication event and therefore the genes can be labeled.

A *duplicated genome* (either labeled or unlabeled) is a genome in which every gene has CN=2. For an ordinary genome $\Pi$ over $\mathcal{G}$, a *doubled genome* $2\Pi = \Pi \cup \Pi$ is an unlabeled duplicated genome over $2\mathcal{G} = \mathcal{G} \cup \mathcal{G}$ in which every gene, adjacency and telomere has two copies. For example, if $\Pi = \{\{a_t\}, \{a_h, b_h\}, \{b_h\}\}$ then $2\Pi = \{\{a_t\}, \{a_h, b_h\}, \{b_h\}, \{a_t\}, \{a_h, b_h\}, \{b_h\}\}$.

The *double distance problem* [2] is defined as follows. Given an ordinary genome $\Pi$ over $\mathcal{G}$, a labeled duplicated genome $\Theta$ and an operation distance measure $d$, find the minimum distance of $\Theta$ to some labeling of $2\Pi$. Formally, the *double distance* between $\Pi$ and $\Theta$ is:

$$dd(\Pi, \Theta) = \min_{\Gamma \in L(2\Pi)} d(\Gamma, \Theta) \tag{1.4}$$

where $L(2\Pi)$ is the set of all possible labelings of $2\Pi$.

The double distance problem can be solved in linear time for the BP [63] and the SCoJ measures [41]. However, it is NP-hard under the DCJ distance [106].

Given a labeled duplicated genome $\Theta$ and an operation distance measure $d$, the *genome halving problem* seeks to find an ordinary genome $\Pi$ that minimizes the double distance to $\Theta$ [38]. Formally, the *halving distance* of $\Theta$ is defined as:

$$hd(\Theta) = \min_{\Pi} dd(\Pi, \Theta) \tag{1.5}$$

The halving distance can be solved in linear time for the BP measure, but if we restrict the genome $\Pi$ to be linear or unichromsomal it becomes NP-hard [63]. For the SCoJ distance, the problem is solvable in linear time even when $\Pi$ is restricted to be a linear or circular genome [41]. Under the DCJ distance the halving problem can be solved in linear time [69, 114] even with Π restricted to a unichromsomal genome [1].

A generalization of the halving problem for finding an ordinary pre-WGD genome given an extant genome with exactly $m > 2$ copies is called *genome aliquoting* [114]. Aliquoting is polynomially solvable for the BP [115] and SCoJ [41] distances, while a 2-approximation algorithm is known for the problem under the DCJ distance [115]. Recently, efficient ILP formulations were suggested for genome halving and aliquoting under the DCJ distance [5].

The *guided genome halving problem* tries to combine both the genome halving and double distance [129]. Given an ordinary genome Δ, a labeled duplicated genome Θ, and an operation distance measure $d$, find an ordinary genome Π that minimizes the sum of the double distance between Π and Θ, plus the distance between Π and Δ. Formally, the guided halving distance is:

$$ghd(\Delta, \Theta) = \min_{\Pi} \left[dd(\Pi, \Theta) + d(\Pi, \Delta)\right] \tag{1.6}$$

The problem can be solved in $O(n^{1.5})$ time for the BP distance, but it becomes NP-hard with additional restrictions [63]. For the SCoJ distance, the problem has linear solutions even with restrictions to linear or circular genomes [41]. It is NP-hard for the DCJ distance [106].

### 1.3.2 Single copy models with indels

Models presented in this section allow new numerical operations while maintaining the assumptions of ordinary genomes. The input is two ordinary genomes Π and Γ over potentially different gene sets $\mathcal{G}_1$ and $\mathcal{G}_2$. The goal is to transform Π into Γ with structural operations and additional operations that introduce new genes or remove genes. All genomes in the sorting scenario must be ordinary. Thus, the use of these models requires to uniquely resolve the homology between the two genomes or introduce exclusive segments.

Given a chromosome string $C = c_1 \ldots c_n$, a *deletion* $del(i, j)$ produces a new chromosome $C[1, i-1] \cdot C[j+1, n]$. An *insertion* $ins(S, i)$ of a sequence $S = s_1 \ldots s_m$ into a chromosome $C$ at position $i$ results in $C[1, i] \cdot S \cdot C[i+1, n]$ (see Figure 1.1). Insertions and deletions are commonly referred to as *indel* operations [22]. Since these models assume that all genomes are ordinary, insertions cannot introduce new copies of genes. Instead, indels are used to add and remove genes that appear in one genome but not in the other.

El-Mabrouk was the first to address sorting permutations by reversals and indels and gave exact an algorithm and a heuristic for specific cases [37]. Improved bound for this problem were later devised [116]. Yancopoulos and Friedberg [119] analyzed the problem of sorting ordinary genomes with DCJs and indels. Their model allowed to insert and delete genes that appear in the source or target genomes, and thus a possible sorting scenario can delete all the chromosomes of the source genome and insert the chromosomes of the target genome. Braga *et al.* [22] gave a linear time algorithm for finding a minimum sorting scenario with DCJs and indels, restricting indels to affect genes that are not common to the source and target genomes. The problem is solvable in linear time even when DCJs and indels have different weights [29].

Braga *et al.* [21] introduced a new operation that generalizes both insertions and deletions. A *substitution* is an operation that replaces a sequence of consecutive genes with another sequence. This operation can be thought of as a deletion of the sequence to be replaced followed by an insertion of the new sequence in the same place. Notice that this operation can implement both deletions and insertions by taking an empty sequence as the new or old sequence, respectively. Sorting ordinary genomes with DCJs and substitutions can be solved in linear time [21], even when substitutions have different weights than DCJs [30].

### 1.3.3 Multi-copy models without duplications/deletions

In this section we focus on comparing genomes with multiple gene copies but without explicit deletion or duplication operations. The comparison can be used to assign orthology relationship between gene copies in the source and target genomes [25]. Given a source genome and a target genome with multiple gene copies, the general approach is to find a matching of the gene copies that minimizes some structural operation distance. Gene copies that are not matched are ignored, so they are implicitly deleted and do not incur the cost of a true deletion operation. Most formulations result in NP-hard problems.

There are three main formulation strategies depending on the cardinality of the

Table 1.1: Multi-copy model results

| Operations | Exemplar | Intermediate | Matching |
|---|---|---|---|
| BP | NP-hard [23]<br>Branch and bound [90, 73]<br>ILP [3, 98] | NP-hard [18]<br>ILP [4, 99]<br>Heuristics [4] | NP-hard [18]<br>Branch and bound [19]<br>ILP [4, 99]<br>Heuristics [4] |
| Reversals and translocations | NP-hard [23] | NP-hard [25]<br>ILP [104]<br>Heuristics [25, 43] | NP-hard [25]<br>ILP [104]<br>Heuristics [25, 43] |
| DCJ | Branch and bound [120] | NP-hard [97] | NP-hard [97]<br>ILP [97]<br>Branch and bound [120]<br>Approximation [96, 87] |

matching of multi-copy genes:

- *Exemplar* strategy [90], in which in each genome, exactly one copy of each gene is selected and all other copies are ignored. It thus assumes that the exemplar is the ancestor that all other gene copies have evolved from.
- *Intermediate* strategy [4], in which the same predefined number of copies (at least one) for each gene are selected and matched between genomes, and all other copies are ignored.
- *Maximum matching* strategy [19], in which for each gene, the maximum possible gene copies (the smaller of the gene's CNs in the two genomes) are selected and matched between genomes, and the remaining copies are ignored.

Although most formulations are NP-hard, several exhaustive and heuristic algorithms have been suggested. In recent years, Integer Linear Programming (ILP) formulations presented by Shao and Moret were used to solve such problems, and have shown good results and scalability [97, 98, 99]. Table 1.1 summarizes selected results for different operations and different formulations.

The majority of hardness results, as well as exact and heuristic algorithms for these problems, originate from the *breakpoint graph decomposition* problem [60, 24].

The goal in this problem is to find a decomposition of a breakpoint graph into a maximum number of edge-disjoint alternating red/blue cycles. A similar maximum cycle decomposition can also be defined for the adjacency graph [96, 97]. Such decomposition induces a matching between genes and the maximum number of cycles minimizes an operation distance measure [96, 97].

### 1.3.4 Models with duplications or deletions

We now describe several models that include deletions or duplications as explicit numerical operations. The goal of all these models is to transform one genome representation into the other with minimum number of structural and numerical operations. Unlike the classical structural operations, numerical operations such as deletions and duplications have no standard definitions.

Chen *et al.* [25] analyzed a model for sorting unlabeled genomes with multiple gene copies using only reversals. Their heuristic, called *SOAR*, was the first method to assign orthology relationship between genes based on not only sequence similarity but also GRs. In a follow-up paper [43], the authors studied a model that allows reversals and single gene duplications. The latter can insert new gene copies at arbitrary positions in the genome. They developed a heuristic called *MSOAR* for matching gene copies between the two input genomes such that the number of reversals plus gene duplications would be minimal. While SOAR requires every gene to have an equal number of copies in the two input genomes, MSOAR alleviates this assumption. In *MSOAR 2.0* [101], only tandem single gene duplications are allowed, and again, an efficient heuristic for this sorting problem is given.

Kahn and Raphael [58] introduced a measure called the *string duplication distance* that models building a target string by repeatedly copying substrings of a fixed source string. The *string duplication* operation, $\delta_{s,t,p}(X)$, copies a substring $x_s \dots x_t$ of string $X$ and pastes it into another string $Z$ at position $p$. Given a source string $X$ without duplicate genes and a target string $Y$ the goal is to find a minimum length sequence of string duplications needed to build the string $Y$. The authors described a polynomial dynamic programming algorithm for computing the distance [58]. In a follow-up work, they enhanced the model to allow substring deletions and inversions. A polynomial dynamic programming algorithm is given for computing the sorting problem [57]. The string duplication model was used for the analysis of repetitive segments in the human genome [56].

A model introduced by Bader [7] allows tandem duplications, segmental deletions and DCJs. Given a labeled chromosome $C$ in string representation, a *tandem duplication* $td(i, j)$ inserts a new copy of the segment $C[i, \dots, j]$ after the $j$'th position, i.e., the new chromosome is $C' = C[1, \dots, i-1] \cdot C[i, \dots, j] \cdot C[i, \dots, j] \cdot C[j+1, \dots, n]$ (Figure 1.1). A *deletion* $del(i, j)$ removes the segment C[i,...,j] and produces $C' = C[1, \dots, i-1] \cdot C[j+1, \dots, n]$ (Figure 1.1). The goal in the model is to find a minimum sorting scenario of the identity chromosome into the input multi copy labeled chromosome. The author gave a lower bound and heuristic for the problem based on the structure of the breakpoint graph.

In a model presented by Shao and Moret [100], labeled genomes are sorted using DCJs and segmental duplications. A *segmental duplication* copies a segment of labeled genes $g_1, \dots, g_m$ of a genome $\Sigma$ and inserts the new labeled copy in $\Sigma$ in a spot outside the original segment. The model allows different costs for different duplications and unit cost for DCJs. However, the optimization problem implicitly assumes that all segmental duplications either precede or follow all DCJ events. Given two labeled genomes $\Pi, \Gamma$, the goal is to find segmental duplications in $\Pi$ and $\Gamma$, remove them, and then find a bijection between the remaining genes such that the cost of segmental duplications plus the DCJ distance is minimized. The authors analyzed this problem and gave an ILP formulation. It is based on the adjacency graph cycle decomposition formulation proposed in [97], applied to a problem instance simplified by detection of optimal substructures.

Paten *et al.* [80] presented a model for genome evolution that does not fit entirely into the standard GR terminology. This model can represent both single base substitutions and structural/numerical rearrangements such as DCJs, deletions and duplications. They defined a data structure called *history graph*, which holds partial order information on the sequence of events. The goal is to find a full sequence of events consistent with the input history graph that minimizes the cost of substitutions and DCJs, while gene deletions and WGDs are free. The authors analyzed this problem and gave polynomially tractable bounds for the cost. In a follow-up paper, Zerbino *et al.* [128] further analyzed the history graph model and showed that the space of possible evolutionary histories can be sampled ergodically.

## 1.4 Multi-copy models in cancer

Cancer is an evolutionary process driven by the accumulation genomic mutations leading to the aberrant function of genes. Those mutations ultimately give cancer cells their proliferative nature. Inferring the evolution of these mutations is an important problem in the research of cancer, both for diagnosis and prognosis [93]. Furthermore, the order in which mutations are acquired can affect disease progression and drug resistance [72], and can identify "driver aberrations" and their order of occurrence in specific cancer types [62].

Cancer genomes are known to undergo structural and numerical changes [47]. These include inversions, chromosomal translocations, tandem duplications, segmental deletions, whole chromosome amplifications or losses and more [112]. Figure 1.4 shows an example of a real cancer karyotype and Figure 1.14 shows a hypothetical sorting scenario for cancer evolution. A large research effort has focused on detecting signatures of these events in tumor genomic data. Currently the effort uses mainly deep sequencing data [32], though traditional methods such as FISH and aCGH are still used to assess the CN of genomic regions. Accurate reconstruction of the numerical and structural variations remains a challenge, and a myriad of computational methods has been devised for this task [32, 107]. Some evolutionary GR models such as those presented in Section 1.3 could also be applied to cancer genomes. Nevertheless, the complexity of tumor karyotypes and their unique characteristics necessitate development of dedicated cancer GR models. For instance, usually different copies of the same segment cannot be distinguished in cancer genomes, and thus they cannot always be labeled.

DCJs can express both reversals and translocations which are frequently observed in cancer genomes and thus can be useful to model tumor evolution. Furthermore, *double minutes*, small circular DNA fragments, have been observed in a large number of human tumors [88], and can be modeled by circular excision operations. However, there is no specific biological evidence that supports the use of the DCJ distance measure.

In Section 1.4.1 we discuss several classical GR models that were applied to cancer data. Section 1.4.2 describes CN edit distance problems in cancer. Section 1.4.3 presents a few other cancer models involving GRs.

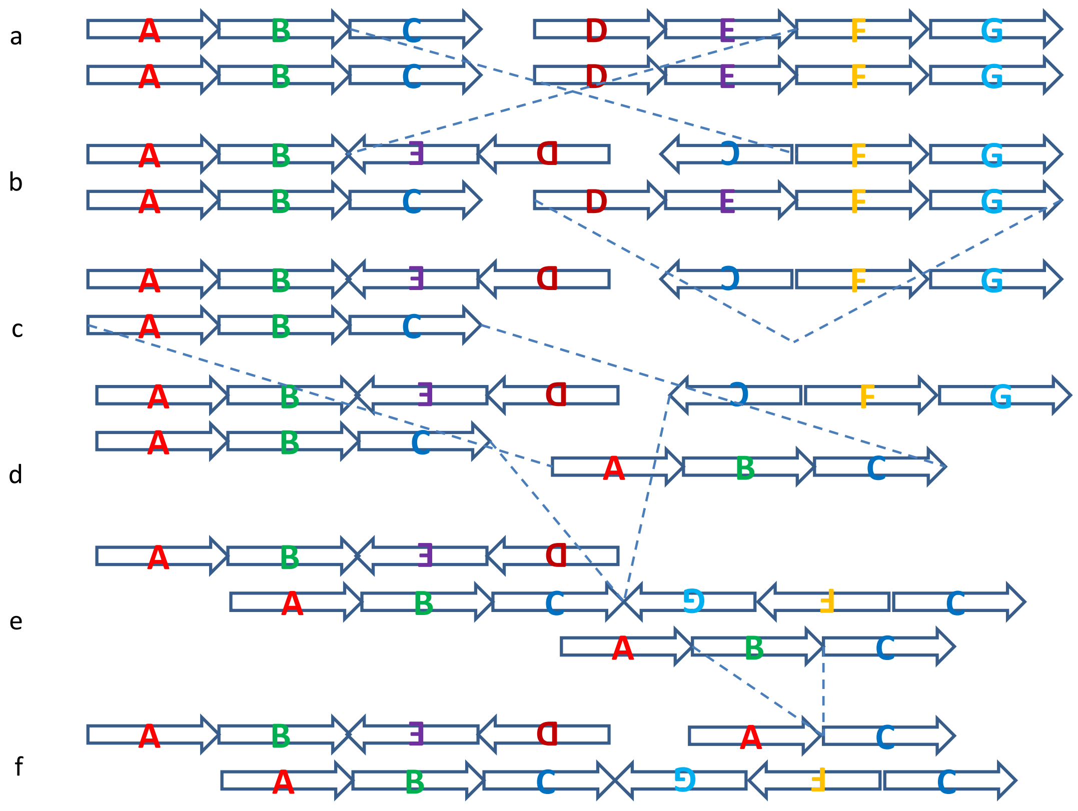

Figure 1.14: A hypothetical sorting scenario for cancer evolution. a. Normal diploid karyotype with two chromosomes. a-b. Translocation. b-c. Chromosome deletion c-d. Chromosome duplication d-e. Fusion e-f. Internal deletion. f. The cancer karyotype. The breakpoints and telomeres involved in each operation are indicated by the broken lines.

### 1.4.1 Models with duplications/deletions

Here we present several GR models with both structural and numerical operations, that were designed to cancer data analysis. All models aim to find a sorting scenario between one genome representation into the another. The source genome is usually the normal genome from a healthy tissue and the target genome is the tumor.

Ozery-Flato and Shamir [78] proposed a GR model designed specifically to analyze chromosomal aberrations in cancer. The inputs for the model are a normal unlabeled source genome with two identical copies of each chromosome and a tumor (target) genome. Both genomes are described as sets of chromosomes, each consisting of a sequence of segments. The goal is to sort the normal genome into the tumor with the fewest cuts, joins, chromosome duplications and chromosome deletions. The authors proved a lower bound for the distance, and presented a polynomial-time 3-approximation algorithm for the problem. They applied the algorithm to over 50,000 low-resolution karyotypes from the Mitelman database [68], which records cancer karyotypes reported in the scientific literature. Interestingly, the approximation algorithm gave an optimal solution in all but 30 karyotypes.

Bader [8] extended his previous model [7] in order to cope with cancer alterations. The revised model accepts multi-chromosomal genomes and allows chromosome deletions and duplications, tandem duplications, segmental deletions and DCJs. A lower bound and a heuristic algorithm were devised, and applied to the Mitelman database [68]. The average calculated distance was 4.08 while the average lower bound was 2.72.

In Chapter 2, we present an analysis of a model for genome sorting using cuts, joins and whole chromosome duplications. A more comprehensive model presented in Chapter 4 accounts for the evolution of unlabeled genomes via DCJs, tandem duplications, segmental deletions, and chromosomal amplifications and deletions.

### 1.4.2 Copy number profile distances

In this section we discuss several models for edit distance between CN profiles. Unlike the genome representations in Section 1.2.1, these profiles give the number of copies of each segment (gene) but do not hold information about their order along the genome. A *copy number profile* (*CNP*) of a chromosome is a vector mapping each gene to a non-negative integer corresponding to the number of copies of the gene in the chromosome. As the order of the genes in a CNP is unknown, it is assumed to be some predefined order (typically the normal genome order). A *genome CNP* is a collection of its chromosome CNPs. We now define operations that transform CNPs and present several models for finding a sorting distance between CNPs.

Let $V = (v_1, \dots, v_n)$ where $v_i \in \mathbb{N} \cup \{0\}$ be a CNP of a chromosome with $n$ genes. A *copy number operation* (*CNO*) is a triple $c = (\ell, h, w)$ where $1 \leq \ell \leq h \leq n$ and $w \in \{1, -1\}$. We say that the operation is a *deletion* if $w = -1$ and an *amplification* if $w = 1$. Applying an operation $c$ to a CNP $V$ results in a new CNP $c(V) = (c(v_1), \dots, c(v_n))$ such that for every $\ell \leq i \leq h$, $v_i > 0$ we have $c(v_i) = v_i + w$, and otherwise $c(v_i) = v_i$. In other words, the operation increases or decreases the

CN of the genes in the interval $[\ell, h]$ if they have a positive CN, while the values of genes outside the interval and zero values are unchanged (see Figure 1.15).

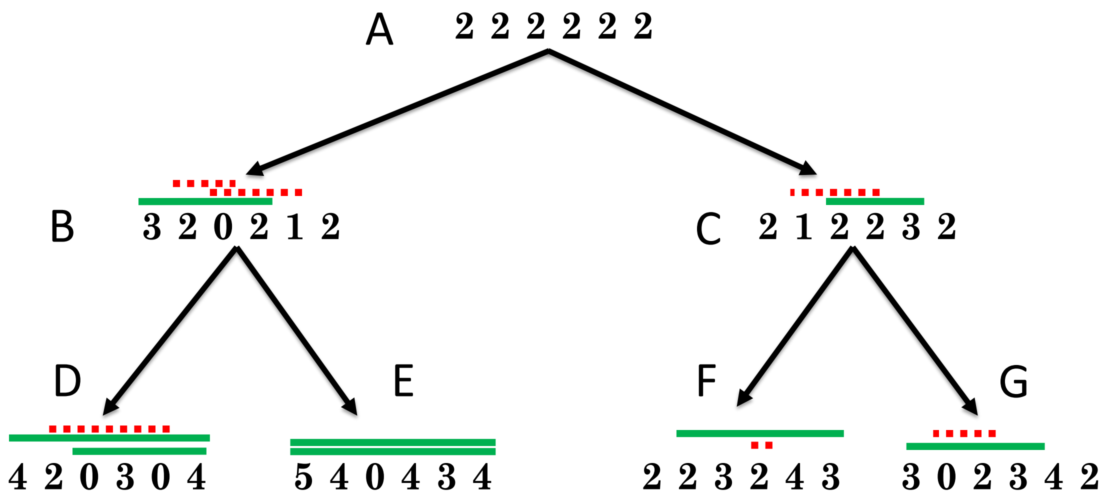

Figure 1.15: Copy number profile evolution. A diploid CNP (A) evolves via CNOs into four extant CNPs (D, E, F, G). Dotted lines represent deletions and bold lines represent amplifications. The order of operations is from top to bottom. For instance, CNP A evolves into CNP B by a deletion of positions 2-3, a deletion of positions 3-5 and an amplification of positions 1-4 (in this order). The corresponding sequence of profiles is 2 2 2 2 2 2 $\rightarrow$ 2 1 1 2 2 2 $\rightarrow$ 2 1 0 1 1 2 $\rightarrow$ 3 2 0 2 1 2. The entire tree has six deletions and eight amplifications.

Chowdhury *et al.* [28] defined edit distance between CNPs obtained from FISH, where the edit operations are amplification or deletion of single genes, single chromosomes, or the whole genome, and presented an algorithm for calculating the distance. The algorithm was exponential in the number of genes and therefore is limited to low-resolution FISH data. An algorithm based on the pairwise distance matrix was used to heuristically infer tumor phylogenies from FISH single cell data. A follow up paper [27] accounted for different weights for different types of operations, again providing an exponential time algorithm.

Schwartz *et al.* [94] introduced a model that admits amplifications and deletions of general contiguous segments in a chromosome CNP. The edit distance between two CNPs is the minimum number of CNOs over all possible separations of the profiles into two alleles. The authors developed an algorithm called *MEDICC* for computing the edit distance, which uses finite-state transducers [70] and is exponential in the maximum CN. MEDICC was used to infer tumor phylogenies from CGH arrays of high grade serous ovarian cancer samples [93].

In Chapter 3, we analyze the problem of sorting one CNP into another using a minimum number of CNOs and give a linear time algorithm to solve it. Notice that this edit distance is not symmetric and in fact there may not be any sequence of CNOs from one given CNP to another since genes with zero copies cannot reappear later in the sequence. To cope with this drawback, we analyzed a symmetric version that given two CNPs aims to find a common ancestor profile that minimizes the sum of distances to these CNPs [36]. We gave a pseudo-polynomial dynamic programming algorithm that is linear in the profile length, and an ILP formulation.

In the more general cancer context, we showed that it is NP-hard to build a phylogenetic tree whose leaves are the input CNPs that minimizes the total number of CNOs along edges in the tree (see Figure 1.15), and gave a practical ILP formulation for this problem. Extending the CNP tree model, Zaccaria *et al.* [121] considered a model in which a fractional (non integer) CNP is allowed, due to the superposition of several CNPs of different subclones. The goal in this case is to deconvolve the fractional CNPs into a weighted sum of integer CNPs such that the phylogenetic tree built over them has minimum CNOs. A heuristic algorithm was given for the problem.

### 1.4.3 Other cancer models

Reconstruction of the exact cancer chromosomes based on short paired-end deep sequencing read data remains a hard challenge. There is a plethora of methods for detection of local rearrangement events and breakpoints [32], but only a few methods try to reconstruct the entire genome. Here we describe a few methods designed for reconstructing cancer genomes. The output genome representation of such methods can be used as input to genome rearrangement models described earlier.

Oesper *et al.* [75] expanded the genome graph into a structure called the *interval adjacency graph*, which represents breakpoints, discordant reads and CN information. Their method, called *PREGO*, uses the number of reads supporting each edge to resolve the CN of genomic segments and identify discordant adjacencies in the tumor genome, and maps this information to the graph. PREGO was shown to efficiently identify complex rearrangements in ovarian cancer data. Eitan and Shamir [35] expanded this model and tested it in extensive simulations and on real cancer data. Their analysis shows that perfect reconstruction of a complete karyotype based on short read data is very hard, but that by several measures, reasonably good reconstructions are obtainable.

*Weaver*, developed by Li *et al.* [65], is a different probabilistic graph model proposed in order to estimate both the CNs and inter-connectivity of SVs. Weaver detects and quantifies CNs and SVs specific for each allele, and was also used for predicting partial timing of SVs relative to chromosome amplifications. A recent expansion of Weaver based on ILP formulation enabled improved prediction of SV phasing and interconnectivity [85].

A probabilistic framework based on breakpoint graphs was presented by Greenman *et al.* [45] for the analysis of mutations and karyotypes from sequencing data. This work tries to reconstruct both the temporal sequence of rearrangements and assemble genomic segments into karyotypes. It uses allelic integer CNs for each segment, the adjacencies between segments and the multiplicity distribution of somatic SNVs. Taking into consideration SNVs can disambiguate some sorting scenarios, since duplicated segments carry the SNVs of the original one. The method can derive partial order of accumulating numerical and single nucleotide mutations. The framework, called *GRAFT*, was demonstrated to work well with a breast cancer sample and cancer cell lines, albeit with limitations imposed by the data quality and the genome complexities.

In contrast to local genome rearrangement operations, *complex genomic rearrangements* (*CGRs*) events are also emerging as a feature of cancer genomes. These events are characterized by multiple genomic breakpoints and fusion, and thus simultaneously affect multiple genes. CGRs involve three or more distant regions of the genome abnormally joining together. nFuse [67] has been designed to discover CGRs in cancer using high-throughput sequencing. The algorithmic method behind nFuse is inspired by DG analysis and is based on shortest alternating paths in breakpoint graphs. CouGaR [34] is a method for characterizing the genomic structure of amplified CGRs, leveraging both depth of coverage and discordant pair-end mapping techniques similar to PREGO [75]. Both methods were successfully applied to cancer genome data and some of their predicted CGRs have been experimentally validated.

## 1.5 Summary of articles included in this thesis

1. **Sorting by cuts, joins and whole chromosome duplications**
   Ron Zeira and Ron Shamir
   Published in *Proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching (CPM 2015)* [124] and *Journal of Computational Biology (JCB)* [125]

   Genome rearrangement problems have been extensively studied due to their importance in biology. Most studied models assumed a single copy per gene. However, in reality duplicated genes are common, most notably in cancer. Here we make a step towards handling duplicated genes by considering a model that allows the atomic operations of cut, join and whole chromosome duplication. Given two linear genomes, $\Gamma$ with one copy per gene, and $\Delta$ with two copies per gene, we give a linear time algorithm for computing a shortest sequence of operations transforming $\Gamma$ into $\Delta$ such that all intermediate genomes are linear. We also show that computing an optimal sequence with fewest duplications is NP-hard.

2. **A linear-time algorithm for the copy number transformation problem**
   Ron Zeira, Meirav Zehavi and Ron Shamir
   Published in *Proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)* [95] and *Journal of Computational Biology (JCB)* [127]

   Problems of genome rearrangement are central in both evolution and cancer. Most evolutionary scenarios have been studied under the assumption that the genome contains a single copy of each gene. In contrast, tumor genomes undergo deletions and duplications, and thus the number of copies of genes varies. The number of copies of each segment along a chromosome is called its *copy number profile*. Understanding copy number profile changes can assist in predicting disease progression and treatment. To date, questions related to distances between copy number profiles gained little scientific attention. Here we focus on the following fundamental problem, introduced by Schwarz *et al.* [94]: given two copy number profiles, $u$ and $v$, compute the minimum number of operations transforming $u$ into $v$, where the edit operations are segmental deletions and amplifications. We establish the computational complexity of this problem, showing that it is solvable in linear time and constant space.

3. **Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions**
   Ron Zeira and Ron Shamir
   Published in *Bioinformatics* [126]

   **Motivation:** Problems of genome rearrangement are central in both evolution and cancer research. Most genome rearrangement models assume that the genome contains a single copy of each gene and the only changes in the genome are structural, i.e., reordering of segments. In contrast, tumor genomes also undergo numerical changes such as deletions and duplications, and thus the number of copies of genes varies. Dealing with unequal gene content is a very challenging task, addressed by few algorithms to date. More realistic models are needed to help trace genome evolution during tumorigenesis.
   **Results:** Here we present a model for the evolution of genomes with multiple gene copies using the operation types double-cut-and-joins, duplications and deletions. The events supported by the model are *reversals*, *translocations*, *tandem duplications*, *segmental deletions*, and *chromosomal amplifications* and *deletions*, covering most types of structural and numerical changes observed in tumor samples. Our goal is to find a series of operations of minimum length that transform one karyotype into the other. We show that the problem is NP-hard and give an integer linear programming formulation that solves the problem exactly under some mild assumptions. We test our method on simulated genomes and on ovarian cancer genomes. Our study advances the state of the art in two ways: It allows a broader set of operations than extant models, thus being more realistic, and it is the first study attempting to reconstruct the full sequence of structural and numerical events during cancer evolution.

# Chapter 2

# Sorting by Cuts, Joins, and Whole Chromosome Duplications

JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 24, Number 2, 2017
© Mary Ann Liebert, Inc.
Pp. 127–137
DOI: 10.1089/cmb.2016.0045

Research Articles

# Sorting by Cuts, Joins, and Whole Chromosome Duplications

RON ZEIRA and RON SHAMIR

## ABSTRACT

**Genome rearrangement problems have been extensively studied due to their importance in biology. Most studied models assumed a single copy per gene. However, in reality, duplicated genes are common, most notably in cancer. In this study, we make a step toward handling duplicated genes by considering a model that allows the atomic operations of cut, join, and whole chromosome duplication. Given two linear genomes, $\Gamma$ with one copy per gene and $\Delta$ with two copies per gene, we give a linear time algorithm for computing a shortest sequence of operations transforming $\Gamma$ into $\Delta$ such that all intermediate genomes are linear. We also show that computing an optimal sequence with fewest duplications is NP-hard.**

**Keywords:** computational genomics, genome rearrangements, SCJ.

## 1. INTRODUCTION

Genome organization evolves over time by undergoing rearrangement operations. Finding a shortest sequence of operations (also called a sorting scenario) between two genomes is the focus of the field of *genome rearrangements*. Such problems were studied extensively over the last two decades due to their importance in evolution (Fertin et al., 2009).

The combinatorial problems in genome rearrangements depend on the allowed operations. Hannenhalli and Pevzner (1995) showed in their seminal work that finding the minimal number of inversions that transform one signed genome into another is polynomial. Many other models were studied later, allowing one or several types of operations (Hannenhalli and Pevzner, 1995; Christie, 1996; Hannenhalli, 1996; Caprara, 1997; Dias and Meidanis, 2001; Lu et al., 2006; Mira and Meidanis, 2007; Bulteau et al., 2012).

The double cut and join (DCJ) operation (Yancopoulos et al., 2005) models reversals, transpositions, translocations, fusions, fissions, and block-interchanges as variations of one basic operation. A DCJ operation cuts the genome in two places, producing four open ends, and rejoins them in two new pairs. Finding the DCJ distance between two gene permutations can be done in linear time (Bergeron et al., 2006). The single cut or join (SCJ) model (Feijão and Meidanis, 2011) further simplifies the model and allows polynomial solutions to some rearrangement problems that are NP-hard under most formulations. An SCJ operation either cuts a chromosome or joins two chromosome ends. This simple model gives good results in real biological applications (Biller et al., 2013).

Models of genomes that assume a single copy of each gene are too restrictive for many real biological problems. Duplications are frequent in cancer genomes, especially in oncogenic regions (Bayani et al.,

Blavatnik School of Computer Science, Tel Aviv University, Tel-Aviv, Israel.

2007). Most plant genomes contain large duplicated segments (Blanc et al., 2000). A major evolutionary event is *whole genome duplication*, wherein all chromosomes are duplicated (Savard et al., 2011).

In spite of their importance, models that allow duplications as rearrangement operations have not been the subject of extensive research to date. Ozery-Flato and Shamir (2009) considered a model that includes certain duplications, deletions, and SCJ operations. Under some simplifying assumptions, they provided a 3-approximation algorithm that performed well on cancer genomes. Bader (2009, 2010) provided a heuristic for sorting by DCJs, duplications, and deletions. Shao et al. (2013) studied sorting genomes using DCJs and segmental duplications and provided an algorithm to improve an initial sorting scenario. The majority of extant models for genomes with multiple gene copies result in NP-hard problems (Tannier et al., 2009; Savard et al., 2011; Shao and Lin, 2012; Shao et al., 2014).

In this article, we present a model that allows the operations, cut, join, and whole chromosome duplication. We call it the *SCJD model*. The model tries to combine the simplicity and applicability of the SCJ model while allowing chromosomal duplications observed in cancer genomes. In fact, aneuploidy (whole chromosome duplication or deletion) is a hallmark of cancer, and how cancer cells can adapt to tolerate aneuploidy remains a key enigma (Gordon et al., 2012; Giam and Rancati, 2015). The SCJD model is applicable for tumor genomes where one copy of each chromosome remains unaffected and the other is duplicated at most once. Thus, we view this model as moving one step closer to a realistic model of cancer genome evolution. Such model should allow multiple gene copies and reversals, translocations, duplications, and deletions.

Given two linear genomes, $\Gamma$ with one copy per gene and $\Delta$ with two copies per gene, we give a linear time algorithm for computing a shortest sequence of operations transforming $\Gamma$ into $\Delta$, where all intermediate genomes must be linear too. We provide a closed-form formula for that sequence length. In addition, we show that there is an optimal sequence in which all duplications are consecutive.

While cuts or joins are local events, a duplication of an entire chromosome is a more drastic event. We show that our algorithm actually gives an optimal scenario with a maximum number of duplications. On the other hand, we prove that finding a conservative optimal SCJD scenario with fewest duplications is NP-hard.

The structure of this article is as follows. We give computational background in Section 2. In Section 3, we present the SCJD model. Section 4 gives the algorithm for the SCJD sorting problem and Section 5 shows the NP-hardness result. Finally, in Section 6, we present a brief discussion and suggest future directions. A preliminary version of this article appeared in the proceedings of CPM 2015 (Zeira and Shamir, 2015).

## 2. PRELIMINARIES

### 2.1. Genome representation

We use the following standard terminology in genome rearrangements (Bergeron et al., 2006). The basic entities are *genes*, denoted by $a$, $b$, $c$ etc. Gene $a$ has *extremities*: a *head* $a_h$ and a *tail* $a_t$. Gene $a$ is assumed to be oriented from its tail to its head and is *positively oriented* if $a_t$ is to the left of $a_h$. A *negatively oriented* gene $a$ is denoted by $-a$. A *chromosome* is a sequence of oriented genes, for example, $C = ab - c - d$. An *adjacency* in a chromosome is a consecutive pair of extremities from distinct neighboring genes. For example, the adjacencies in $C$ above are $\{a_h, b_t\}$, $\{b_h, c_h\}$, and $\{c_t, d_h\}$. A *telomere* is an extremity that is not adjacent to any other gene, corresponding to the end of a chromosome, for example, $\{a_t\}$,$\{d_t\}$ in $C$. Hence, a chromosome can be equivalently represented by its set of adjacencies, where the telomeres are implicit. Note that the set of adjacencies defining a chromosome is identical to that of the reverse chromosome, where order and orientation of genes are inverted (the reverse of $C$ is $-C = dc - b - a$). Hence, a chromosome and its reverse are equivalent.

A *genome* over gene set $\mathcal{G}$ is a collection of chromosomes. We assume for now that each gene appears once, for example, $\Gamma = \{ab, c - d\}$. Equivalently, it can be defined by a set of adjacencies such that for each gene in $\mathcal{G}$, each extremity appears at most once. Hence, $\Gamma = \{\{a_h, b_t\}, \{c_h, d_h\}\}$. The *size* of a genome $\Gamma$, denoted by $|\Gamma|$, is the number of adjacencies in it. A chromosome is called *linear* if it starts and ends with a telomere and *circular* if it does not contain any telomere, for example, $D = \{\{a_h, b_t\}, \{b_h, a_t\}\}$. For a sequence of genes $S$, denote by $S$ and $(S)$ the corresponding linear and circular chromosomes, respectively. For example, the linear chromosome $a - b$ is defined by the set of adjacencies $\{\{a_h, b_h\}\}$ and the circular chromosome $(a - b)$ is defined by the set $\{\{a_h, b_h\}, \{b_t, a_t\}\}$. A genome is called *linear* if all its chromosomes are linear.

A gene that has several copies in the genome is called *duplicated*. We *label* different copies of the same gene by superscripts, for example, copies $a^1$ and $a^2$ of gene $a$. A *duplicated genome* has exactly two copies of each gene. A genome with a single copy of each gene is called *ordinary*. The duplication of an ordinary genome $\Pi$ creates a special kind of genome (Warren and Sankoff, 2011): each gene and each adjacency in $\Pi$ is doubled, producing the genome $\Pi \oplus \Pi$. Note that in $\Pi \oplus \Pi$, the two copies of each gene are unlabeled. The set of all possible labeled genomes corresponding to $\Pi \oplus \Pi$ is denoted by $2\Pi$. A genome $\Sigma \in 2\Pi$ is called a *perfectly duplicated genome*. Hence, for $\Gamma$ above, $\Gamma \oplus \Gamma = \{ab, ab, c-d, c-d\}$ and $\Sigma = \{\{a_h^2, b_t^2\}, \{c_h^2, d_h^1\}, \{a_h^1, b_t^1\}, \{c_h^1, d_h^2\}\} \in 2\Gamma$.

## 2.2. SCJ distance

A *cut* operation takes an adjacency $\{x, y\}$ and breaks it into two telomeres, $\{x\}$ and $\{y\}$. The reverse operation, called a *join*, combines two telomeres, $\{x\}$ and $\{y\}$, into an adjacency $\{x, y\}$. An SCJ operation is either a cut or a join (Feijão and Meidanis, 2011). Given two ordinary genomes, $\Pi$ and $\Gamma$, on the same gene set, a sequence of SCJ operations that transforms $\Pi$ into $\Gamma$ is called a *sorting scenario*. The *SCJ distance*, denoted by $d_{SCJ}(\Pi, \Gamma)$, is the length of a shortest sorting scenario between $\Pi$ and $\Gamma$. Feijão and Meidanis give the following solution for the SCJ distance:

**Theorem 1.** *(Feijão and Meidanis, 2011)* $d_{SCJ}(\Pi, \Gamma) = |\Pi \backslash \Gamma| + |\Gamma \backslash \Pi| = |\Pi| + |\Gamma| - 2|\Pi \cap \Gamma|$. $\Pi \backslash \Gamma$ *defines the set of cuts and* $\Gamma \backslash \Pi$ *defines the set of joins in an optimal sorting scenario.*

## 2.3. Double distance

The *SCJ double distance* between an ordinary genome, $\Gamma$, and a duplicated genome, $\Delta$, is defined as

$$dd_{SCJ}(\Gamma, \Delta) \equiv \min_{\Sigma \in 2\Gamma} d_{SCJ}(\Sigma, \Delta) \tag{1}$$

Hence, in the *double distance problem*, one seeks a labeling of each gene copy in a perfectly duplicated genome $\Sigma \in 2\Gamma$ that minimizes the SCJ distance to $\Delta$.

For a genome $\Sigma$ and an adjacency $\alpha = \{x, y\}$, let $\Sigma_\alpha$ be the set of all adjacencies of the form $\{x^i, y^j\}$ in $\Sigma$. Hence, $|\Sigma_\alpha|$ can be 0, 1, or 2 if $\Sigma$ is duplicated and 0 or 1 if $\Sigma$ is ordinary. Let $A = \{\alpha = \{x, y\} | x \neq y\}$ be the set of all possible adjacencies with extremities belonging to distinct genes. A solution to the double distance problem is given by the following theorem:

**Theorem 2.** *(Feijão and Meidanis, 2011) The SCJ double distance between an ordinary genome, $\Gamma$, and a duplicated genome, $\Delta$, is*

$$dd_{SCJ}(\Gamma, \Delta) = |\Delta| + 2\sum_{\alpha \in A} |\Gamma_\alpha|(1 - |\Delta_\alpha|).$$

*A perfectly duplicated genome, $\Sigma \in 2\Gamma$, realizing the distance is obtained by taking for each adjacency $\alpha = \{x, y\} \in \Gamma$ (1) the labeled adjacencies of $\Delta_\alpha$ and (2) adjacencies $\{x^i, y^j\}$ with arbitrary labeling that do not conflict with (1) or among themselves.*

## 3. THE SCJD MODEL

In this section, we generalize the SCJ model to allow duplications.

A *duplication* operation on a genome, $\Pi$, takes a linear chromosome, $C$, in $\Pi$ and produces a new genome, $\Pi'$, with an additional copy of the chromosome. For example, if $\Pi = \{abcd, efg\}$, then a duplication of the first chromosome will give $\Pi' = \{abcd, abcd, efg\}$. An *SCJD operation* is either an SCJ or a duplication.

Given two linear genomes on the same gene set of size $n$, an ordinary one, $\Gamma$, and a duplicated one, $\Delta$, a sequence of SCJD operations that transforms $\Gamma$ into $\Delta$ is called an *SCJD sorting scenario*. The *SCJD distance*, denoted by $d_{SCJD}(\Gamma, \Delta)$, is the number of operations in a shortest SCJD sorting scenario between $\Gamma$ and $\Delta$.

Since we focus on linear genomes, we will assume from now on that all chromosomes, including intermediate ones, are linear unless specified otherwise. The following simple lemma shows that this can be satisfied when using only SCJ operations:

**Lemma 1**. *A sequence of SCJ operations transforming one linear genome into another linear genome can be reordered, producing another sequence with the same length such that all intermediate genomes are linear.*

*Proof.* Let $\Pi$ and $\Gamma$ be two ordinary linear genomes and let $o_1, \ldots, o_d$ be a sequence of SCJ operations transforming $\Pi$ into $\Gamma$. Suppose there is a join operation, $o_i$, which creates a circular chromosome $C$ by joining its two telomeres, $x$ and $y$. Since $\Gamma$ is linear, there is a cut operation $o_i$ for $i < j$ that breaks $C$ into a linear chromosome by cutting an adjacency $\{w, z\}$. We create a new sequence of the same length by replacing $o_i$ with a cut of $\{w, z\}$ and replacing $o_j$ with a join of $x$ and $y$. In the new sequence, the chromosome $C$ is linear. By repeating the argument for every intermediate circular chromosome, the lemma holds. ■

The examples below demonstrate SCJ double distances and SCJD sorting scenarios. For simplicity, we drop the braces around genomes from now on.

*Example 1.* $\Gamma = a$, $\Delta = a - a$; $dd_{SCJ}(\Gamma, \Delta) = 1$; $d_{SCJD}(\Gamma, \Delta) = 2$:

$$\Gamma \underset{dup}{\rightarrow} a, a \underset{join}{\rightarrow} \Delta$$

*Example 2.* $\Gamma = ab$, $\Delta = ab, ab$; $dd_{SCJ}(\Gamma, \Delta) = 0$; $d_{SCJD}(\Gamma, \Delta) = 1$:

$$\Gamma \underset{dup}{\rightarrow} \Delta$$

*Example 3.* $\Gamma = a, bc$, $\Delta = ab, abcc$; $dd_{SCJ}(\Gamma, \Delta) = 4$; $d_{SCJD}(\Gamma, \Delta) \leq 4$:

$$\Gamma \underset{join}{\rightarrow} abc \underset{dup}{\rightarrow} abc, abc \underset{cut}{\rightarrow} abc, ab, c \underset{join}{\rightarrow} \Delta$$

*Example 4.* $\Gamma = acb$, $\Delta = abab, cc$; $dd_{SCJ}(\Gamma, \Delta) = 8$; $d_{SCJD}(\Gamma, \Delta) \leq 7$:

$$\Gamma \underset{cut}{\rightarrow} a, cb \underset{cut}{\rightarrow} a, b, c \underset{join}{\rightarrow} ab, c \underset{dup}{\rightarrow} ab, ab, c \underset{dup}{\rightarrow} ab, ab, c, c \underset{join}{\rightarrow} abab, c, c \underset{join}{\rightarrow} \Delta$$

Let $\#_c\Pi$ be the number of linear chromosomes in genome $\Pi$. Let $\Gamma$ be an ordinary linear genome and let $\Delta$ be a duplicated linear genome on the same gene set. A trivial upper bound for the SCJD distance between $\Gamma$ and $\Delta$ is given by solving the double distance between $\Delta$ and $\Gamma$. This corresponds to first duplicating each chromosome in $\Gamma$ and then computing the SCJ distance between $\Delta$ and $\Gamma \oplus \Gamma$. We get $d_{SCJD}(\Gamma, \Delta) \leq dd_{SCJ}(\Gamma, \Delta) + \#_c\Gamma$. However, Example 3 shows that this bound is not tight. It is tempting to guess that $dd_{SCJ}(\Gamma, \Delta) \leq d_{SCJD}(\Gamma, \Delta)$. Alas, Example 4 shows this conjecture is incorrect.

## 4. COMPUTING THE SCJD DISTANCE

In this section, we will solve the SCJD distance problem. The key idea is to show that there is an optimal scenario in which all the duplication operations are performed in sequence, one after the other. Having shown that, the sorting scenario between $\Gamma$ and $\Delta$ can be presented as follows:

1. Transform $\Gamma$ into another ordinary linear genome, $\Gamma'$, using only SCJ operations.
2. Duplicate all the chromosomes of $\Gamma'$ resulting in a duplicated genome, $\Gamma' \oplus \Gamma'$.
3. Solve the SCJ double distance problem between $\Gamma'$ and $\Delta$.

Let $O^* = o_1, \ldots, o_d$ be an optimal SCJD sorting scenario. Let $\Gamma_0 \equiv \Gamma$ and for every $1 \leq i \leq d$, let $\Gamma_i = o_i(\Gamma_{i-1})$ be the genome resulting from performing $o_i$ on $\Gamma_{i-1}$. By definition, $\Gamma_d \equiv \Delta$. Let $D_i$ be the set of duplicated genes in $\Gamma_i$. We have $D_0 = \emptyset$ and $D_d = \mathcal{G}$. Given a gene set $\mathcal{H}$, denote its extremity set by $\mathcal{E}_{\mathcal{H}} = \{a_t | a \in \mathcal{H}\} \cup \{a_h | a \in \mathcal{H}\}$.

**Proposition 1.** *In an optimal sorting scenario $O^*$, if $o_i$ is a join operation acting on the two telomeres, $x$ and $y$, then either both $x, y \in \mathcal{E}_{D_i}$ or both $x, y \notin \mathcal{E}_{D_i}$.*

*Proof.* Since $o_i$ is not a duplication, we have $D_{i-1} = D_i$. Suppose by contradiction that $x \in \mathcal{E}_{D_i}$, but $y \notin \mathcal{E}_{D_i}$. Let $o_j$ $(i < j)$ be the first duplication such that $y \in \mathcal{E}_{D_j}$. The duplication operation must act on a chromosome in which all genes are not yet duplicated. Therefore, there is a cut operation $o_k (i < k < j)$ that breaks the adjacency $\{x, y\}$ created by $o_i$.

Let $O' = o'_1, \ldots, o'_{d-2} = o_1, \ldots, o_{i-1}, o_{i+1}, \ldots, o_{k-1}, o_{k+1}, \ldots, o_d$ be an alternative sorting sequence that results from removing $o_i$ and $o_k$ from $O^*$. Let $\Gamma'_0 \equiv \Gamma$, and denote $\Gamma'_l = o'_l(\Gamma'_{l-1})$. For every $l$ with $1 \leq l \leq i-1$, by definition, $o'_l = o_l$ and therefore $\Gamma'_l = \Gamma_l$.

We first show that for every $l$ with $i \leq l \leq k-2$, $\Gamma'_l = \Gamma_{l+1} \backslash \{\{x, y\}\}$. Since $o_i$ creates the adjacency $\{x, y\}$, we have $\Gamma_i = \Gamma_{i-1} \cup \{\{x, y\}\}$. For every such $l$, $o'_l = o_{l+1}$ and since none of these operations creates a new copy of $y$, we have $\Gamma'_l = \Gamma_{l+1} \backslash \{\{x, y\}\}$.

Next, we show that for every $l$ with $k-1 \leq l \leq d-2$, $\Gamma'_l = \Gamma_{l+2}$. From the previous result and the fact that $\Gamma_k = \Gamma_{k-1} \backslash \{\{x, y\}\}$, we have $\Gamma'_{k-2} = \Gamma_k$. Now, for every such $l$, $o'_l = o_{l+2}$ and therefore $\Gamma'_l = \Gamma_{l+2}$.

We have established that $O'$ is an SCJD sorting sequence of length, $d-2$, contradicting the optimality of $O^*$. ■

**Proposition 2.** *In an optimal sorting scenario $O^*$, if $o_i$ is a cut operation acting on the adjacency $\{x, y\}$, then either $x, y \in \mathcal{E}_{D_i}$* or *$x, y \notin \mathcal{E}_{D_i}$.*

*Proof.* Suppose by contradiction, $x \in \mathcal{E}_{D_i}$, but $y \notin \mathcal{E}_{D_i}$. Let $o_j$ $(j < i)$ be the first duplication such that $x \in \mathcal{E}_{D_j}$. Since $x$ was duplicated as part of a chromosome that did not contain $y$, there is subsequently a join operation $o_k$ $(j < k < i)$ that creates the adjacency $\{x, y\}$. Defining $O' = o'_1, \ldots, o'_{d-2} = o_1, \ldots, o_{k-1}, o_{k+1}, \ldots, o_{i-1}, o_{i+1}, \ldots, o_d$, we can get a shorter SCJD sorting scenario in a similar manner as the proof of Proposition 1. ■

**Corollary 1.** *In an optimal sequence of SCJD operations, at the time of a cut or a join operation on the two extremities, x and y, either the genes corresponding to both x and y have already been duplicated or none of them have.* ■

We say that a join operation in a sorting scenario is *valid* only if the two extremities it joins are not already part of any other adjacency. Similarly, a cut operation is *valid* only if the adjacency it breaks exists. A duplication operation is *valid* only if it duplicates a linear chromosome such that all its genes were not previously duplicated. A sorting scenario is *valid* if all its operations are valid.

Let $S = s_1, \ldots, s_m$ be a valid SCJD sorting scenario between $\Gamma$ and $\Delta$. We say the operation, $s_{i+1}$, can *preempt* the operation, $s_i$, if the sequence, $S' = s_1, \ldots, s_{i+1}, s_i, \ldots, s_m$, is also a valid SCJD sorting scenario between $\Gamma$ and $\Delta$.

**Proposition 3.** *In a valid SCJD scenario, S transforming $\Gamma$ into $\Delta$, if $s_{i+1}$ is an SCJ operation acting on two extremities $x, y$ that were not duplicated and $s_i$ is a duplication, then $s_{i+1}$ can preempt $s_i$.*

*Proof.* Suppose $s_i$ duplicates the linear chromosome $C$ and produces another copy of it $C'$. Since $s_{i+1}$ operates on genes that are not duplicated yet, none of those genes belong to $C$ or $C'$. Therefore, the sequence, $s_1, \ldots, s_{i-1}, s_{i+1}$, is valid. Any operation that creates an adjacency or a telomere of $C$ must precede $s_i$. Hence, $s_1, \ldots, s_{i-1}, s_{i+1}, s_i$ is valid. Finally, any $s_j$ for $j > i+1$ that requires the results of $s_i$ or $s_{i+1}$ is still valid. Thus, $S' = s_1, \ldots, s_{i-1}, s_{i+1}, s_i, s_{i+2}, \ldots, s_m$ is a valid sequence.

To conclude the proof, we need to show that $\Gamma_{i+1} \equiv \Gamma'_{i+1}$. Indeed, $s_{i+1}$ does not alter any of the adjacencies or telomeres of $C$ or $C'$, and therefore, $\Gamma_{i+1} = s_{i+1}(\Gamma_{i-1} \cup C') \equiv s_{i+1}(\Gamma_{i-1}) \cup C' = \Gamma'_{i+1}$. ■

**Proposition 4.** *In a valid SCJD scenario, S transforming $\Gamma$ into $\Delta$, if $s_{i+1}$ is a duplication and $s_i$ is a cut or join acting on two duplicated extremities, then $s_{i+1}$ can preempt $s_i$.*

*Proof.* Suppose $s_i$ is an SCJ operation acting on the two extremities, $x$ and $y$, such that the genes corresponding have both already been duplicated. Let $s_{i+1}$ be a duplication operation that takes the linear chromosome, $C$, and produces another copy of it, $C'$.

Since $S$ is a valid sorting sequence, duplication operations that act on the genes corresponding to $x$ and $y$ must precede $s_i$. In addition, none of these genes are in $C$. Thus, the sequence, $s_1, \ldots, s_{i-1}, s_{i+1}, s_i$, is still valid. Any subsequent operation has its required set of adjacencies and telomeres and thus the sequence $S' = s_1, \ldots, s_{i-1}, s_{i+1}, s_i, s_{i+2}, \ldots, s_m$ is valid. In addition, for the same reasons, $\Gamma_{i+1} = s_i(\Gamma_{i-1}) \cup C' \equiv s_i(\Gamma_{i-1} \cup C') = \Gamma'_{i+1}$. ■

**Proposition 5.** *In a valid SCJD scenario, S transforming $\Gamma$ into $\Delta$, if $s_{i+1}$ is an SCJ acting on two extremities that were not duplicated yet and $s_i$ is an SCJ acting on two duplicated extremities, then $s_{i+1}$ can preempt $s_i$.*

*Proof.* If $\alpha \in \Gamma$, then $\alpha' \notin \Gamma$ since $\Gamma$ is an ordinary genome. In addition, since $\eta(\alpha) > 0$, $|\Delta_\alpha| > 0$, therefore $|\Delta_{\alpha'}| < 2$. Thus, $\eta(\alpha') \leq 0$. If $\alpha \notin \Gamma$, then to achieve $\eta(\alpha) > 0$, we must have $|\Delta_\alpha| = 2$, so in particular $|\Delta_{\alpha'}| = 0$ and therefore $\eta(\alpha') \leq 0$. ■

For a sequence of SCJ operations, $S$, let $S^D$ ($\overline{S}^D$, respectively) be the subsequence of operations that act on two extremities of genes that have (have not, respectively) already been duplicated at the time of the operation. By Corollary 1, for optimal $S$, $\overline{S}^D$ is indeed the complement of $S^D$.

**Proposition 6.** *There exists an optimal sorting scenario in which all duplication events are consecutive.*

*Proof.* Let $o_{i_1}, \ldots, o_{i_p}$ be the duplication events in an optimal sorting scenario. Denote by $S_{i_j}$ the sequence of SCJ operations occurring between the duplications, $o_{i_j}$ and $o_{i_{j+1}}$. In addition, denote by $S_{i_0}$ and $S_{i_p}$ the sequence of SCJ operations before the first duplication and after the last duplication, respectively. Given an optimal scenario, $O^* = S_{i_0}, o_{i_1}, S_{i_1}, o_{i_2}, S_{i_2}, \ldots, S_{i_{p-1}}, o_{i_p}, S_{i_p}$, we modify it into a new sorting scenario $O'$ as follows: using Propositions 3 and 5, preempt SCJ operations acting on unduplicated genes. Using Proposition 4, preempt duplication events. These steps are iterated until no preemption is possible. We get that $O' = S_{i_0}, \overline{S}^D_{i_1}, \ldots, \overline{S}^D_{i_p}, o_{i_1}, \ldots, o_{i_p}, S^D_{i_1}, \ldots, S^D_{i_{p-1}}, S_{i_p}$ is a valid SCJD optimal sequence in which all duplications are consecutive. ■

**Corollary 2.** *There exists an optimal SCJD sorting scenario, consisting, in this order, of (1) SCJ operations on single-copy genes, (2) duplications, and (3) SCJ operations acting on duplicated genes.* ■

Denote by $\Gamma'$ the intermediate (ordinary) genome after step (1). Then, we can conclude the following:

**Theorem 3.** $d_{SCJD}(\Gamma, \Delta) = \min_{\Gamma'}(d_{SCJ}(\Gamma, \Gamma') + \#_c\Gamma' + dd_{SCJ}(\Gamma', \Delta))$ ■

Recall that $n$ is the number of genes in $\Gamma$. Using Theorems 1 and 2 and the fact that $\#_c\Pi = n - |\Pi|$, the distance formula can be simplified as follows:

$$\begin{aligned} d_{SCJD} &= \min_{\Gamma'}\left(|\Gamma| + |\Gamma'| - 2|\Gamma \cap \Gamma'| + n - |\Gamma'| + |\Delta| + 2\sum_{\alpha \in A} |\Gamma'_\alpha|(1 - |\Delta_\alpha|)\right) \\ &= n + |\Delta| + |\Gamma| - 2\max_{\Gamma'}\left(|\Gamma \cap \Gamma'| + \sum_{\alpha \in A} |\Gamma'_\alpha|(|\Delta_\alpha| - 1)\right) \\ &= n + |\Delta| + |\Gamma| - 2\max_{\Gamma'} \sum_{\alpha \in \Gamma'} (|\Gamma_\alpha| + |\Delta_\alpha| - 1) \\ &= n + |\Delta| + |\Gamma| - 2\max_{\Gamma'} \sum_{\alpha \in \Gamma'} \eta(\alpha) = n + |\Delta| + |\Gamma| - 2\max_{\Gamma'} H(\Gamma') \end{aligned} \qquad (2)$$

where $\eta(\alpha) = \eta(\alpha, \Gamma, \Delta) = |\Gamma_\alpha| + |\Delta_\alpha| - 1$ and $H(\Gamma') = \sum_{\alpha \in \Gamma'} \eta(\alpha)$. Intuitively, $H(\Gamma')$ measures the similarity of $\Gamma'$ to $\Gamma$ and $\Delta$ in terms of adjacencies. Since we want to maximize $H(\Gamma')$, we will focus on adjacencies with positive contribution in Equation (2).

**Lemma 2.** *Let $\alpha = \{x, y\}$ be an adjacency such that $\eta(\alpha) > 0$. Then, for every extremity $z \neq y$, the conflicting adjacency $\alpha' = \{x, z\}$ has $\eta(\alpha') \leq 0$.*

*Proof.* If $\alpha \in \Gamma$, then $\alpha' \notin \Gamma$ since $\Gamma$ is an ordinary genome. In addition, since $\eta(\alpha) > 0$, $|\Delta_\alpha| > 0$, therefore $|\Delta_{\alpha'}| < 2$. Thus, $\eta(\alpha') \leq 0$. If $\alpha \notin \Gamma$, then to achieve $\eta(\alpha) > 0$, we must have $|\Delta_\alpha| = 2$, so in particular $|\Delta_{\alpha'}| = 0$ and therefore $\eta(\alpha') \leq 0$. ■

Combining Lemma 2 and Theorem 3, we get a closed formula for the SCJD distance:

**Theorem 4.** *The genome $\Gamma' = \{\alpha = \{x, y\} | \eta(\alpha) > 0\}$ minimizes Equation (2). If $\Gamma'$ is a linear genome, then the SCJD distance is given by $d_{SCJD}(\Gamma, \Delta) = n + |\Delta| + |\Gamma| - 2H(\Gamma')$.* ■

Let us return to the examples in Section 3:

– Example 1: $n = 1$, $|\Delta| = 1$, $|\Gamma| = 0$, $\Gamma' = \emptyset$, $H(\Gamma') = 0 \rightarrow d = 1 + 1 + 0 - 2 * 0 = 2$
– Example 2: $n = 2$, $|\Delta| = 2$, $|\Gamma| = 1$, $\Gamma' = \{\{a_h, b_t\}\}$, $H(\Gamma') = 2 \rightarrow d = 2 + 2 + 1 - 2 * 2 = 1$
– Example 3: $n = 3$, $|\Delta| = 4$, $|\Gamma| = 1$, $\Gamma' = \{\{a_h, b_t\}, \{b_h, c_t\}\}$, $H(\Gamma') = 1 + 1 \rightarrow d = 3 + 4 + 1 - 2 * 2 = 4$
– Example 4: $n = 3$, $|\Delta| = 4$, $|\Gamma| = 2$, $\Gamma' = \{\{a_h, b_t\}\}$, $H(\Gamma') = 1 \rightarrow d = 3 + 4 + 2 - 2 * 1 = 7$

*Example 5.* *$\Gamma = abc$ and $\Delta = cab, bca$. According to Theorem 4, we get $\Gamma' = (abc)$ because $\eta(\{a_h, b_t\}) = \eta(\{b_h, c_t\}) = \eta(\{c_h, a_t\}) = 1$. The corresponding distance is $d = 3$, providing the following invalid sorting scenario:*

$$\Gamma \underset{join}{\rightarrow} (abc) \underset{dup^*}{\rightarrow} (abc), (abc) \underset{cut}{\rightarrow} cab, (abc) \underset{cut}{\rightarrow} \Delta$$

$dup^*$ indicates a duplication of a circular chromosome, an operation that is not allowed in the SCJD model (and has no cost). It is not difficult to verify that there is no valid sorting scenario with $d \leq 3$.

The reason for the discrepancy in Example 5 is that $\#_c(\Gamma') = n - |\Gamma'| = 0$ is not equal to the number of duplications if there are circular chromosomes. Therefore, to minimize the SCJD distance given by Equation (2), we need to maximize $H(\Gamma')$ under the constraint that $\Gamma'$ is a linear genome, that is, $H(\Gamma') \geq H(\tilde{\Gamma})$ for every linear genome $\tilde{\Gamma}$. Lemma 3 shows that we can do so simply by removing one adjacency with $\eta = 1$ from each circular chromosome in $\Gamma'$ and that such adjacency must exist.

**Lemma 3.** *Let $\Gamma' = \{\alpha = \{x, y\} | \eta(\alpha) > 0\}$ and let $\Gamma''$ be a genome obtained by removing one adjacency $\alpha$ with $\eta(\alpha) = 1$ from each circular chromosome in $\Gamma'$. Then, $\Gamma''$ is a linear genome that maximizes $H(\cdot)$ and the SCJD distance is given by $d_{SCJD}(\Delta, \Gamma) = n + |\Delta| + |\Gamma| - 2H(\Gamma'')$.*

*Proof.* Let $\Pi$ be an ordinary linear genome maximizing $H(\Pi)$. From the maximality of $H(\Pi)$, we may assume w.l.o.g. that $\Pi$ does not contain adjacencies with $\eta(\alpha) \leq 0$. By the definition of $\Gamma'$ and from Lemma 2, we have that $\forall \alpha \in \Pi$ if $\eta(\alpha) > 0$, then $\alpha \in \Gamma'$. It follows that $\Pi \subseteq \Gamma'$. Therefore, every linear chromosome in $\Gamma'$ is also in $\Pi$.

Since $\Pi \subseteq \Gamma'$, any linear chromosome $C$ in $\Pi$ that is not in $\Gamma'$ must be fully contained in a circular chromosome $C'$ of $\Gamma'$. From the maximality of $\Pi$, $C$ must contain all adjacencies in $C'$ except for one adjacency $\alpha$ with minimum $\eta(\alpha)$.

Since $\Gamma'$ contains only adjacencies with $\eta(\alpha) = 1$ or 2, the minimal value is an adjacency with $\eta(\alpha) = 1$ if one exists. If $C'$ contains only adjacencies with $\eta(\alpha) = 2$, it follows that $|\Gamma_\alpha| = 1$ for every $\alpha \in C'$, so $\Gamma$ contains the circular chromosome $C'$, contradicting the linearity of $\Gamma$. Hence, $C'$ must contain at least one adjacency with $\eta(\alpha) = 1$. ■

Applying Lemma 3 to Example 5, we get $\Gamma'' = abc$ and $d = 5$:

$$\Gamma \underset{dup}{\rightarrow} abc, abc \underset{cut}{\rightarrow} a, bc, abc \underset{join}{\rightarrow} bca, abc \underset{cut}{\rightarrow} bca, ab, c \underset{join}{\rightarrow} \Delta$$

Instead, we can choose $\Gamma'' = cab$, which gives a different optimal sorting scenario:

$$\Gamma \underset{cut}{\rightarrow} ab, c \underset{join}{\rightarrow} cab \underset{dup}{\rightarrow} cab, cab \underset{cut}{\rightarrow} cab, a, bc \underset{join}{\rightarrow} \Delta$$

Algorithm 1 gives the full procedure for solving the SCJD distance and sorting problems. Each step of the algorithm takes $O(|\Gamma| + |\Delta|)$ time. In conclusion:

**Algorithm 1 SCJD distance**.
**Input**: An ordinary genome, $\Gamma$, and a duplicated genome, $\Delta$, (both linear) on the same gene set.
**Output**: The SCJD distance, $d_{SCJD}(\Gamma, \Delta)$, and an optimal sorting scenario, $o_1, \ldots, o_d$, in which all intermediate genomes are linear.

```
1: $\Gamma' \leftarrow \{\alpha = \{x, y\} | \eta(\alpha) > 0\}$ (Theorem 4)
2: Create a linear genome, $\Gamma''$, by removing one adjacency, $\alpha$, with $\eta(\alpha) = 1$ from each circular chromosome in $\Gamma'$
   (Lemma 3)
3: $d_{SCJD}(\Gamma, \Delta) \leftarrow n + |\Delta| + |\Gamma| - 2H(\Gamma'')$ (Theorem 4, Lemma 3)
4: $o_1, \ldots, o_i \leftarrow$ Sort $\Gamma$ into $\Gamma''$ (Theorem 1, Lemma 1)
5: $o_{i+1}, \ldots, o_j \leftarrow$ Duplicate all chromosomes in $\Gamma''$.
6: $o_{j+1}, \ldots, o_d \leftarrow$ Sort $2\Gamma''$ into $\Delta$ (Theorem 2, Lemma 1).
7: return $d, \overrightarrow{o}$
```

**Theorem 5.** *Algorithm 1 computes the SCJD distance in linear time.* ■

## 5. CONTROLLING THE NUMBER OF DUPLICATIONS

In this section, we discuss how to control the number of duplications in an optimal SCJD sequence. Since the number of duplications is $n-|\Gamma''|$, selecting different intermediate genomes $\Gamma''$ that preserve the SCJD distance can produce scenarios with different number of duplications.

An optimal SCJD scenario with fewer duplications can be viewed as more conservative. The assumption behind this is that duplications are more radical events than breakage (cut) or fusion (join), which are local events.

**Lemma 4.** *Algorithm 1 gives an optimal sorting scenario with a maximum number of duplications.*

*Proof.* Observe first that for any sorting scenario (optimal or suboptimal) transforming $\Gamma$ into $\Delta$, we can assume w.l.o.g. that all duplications are consecutive without affecting the number of operations (Corollary 2). Call the genome right before the duplications the *last ordinary genome*. Denote by $d(\Gamma, \Pi, \Delta)$ the shortest scenario transforming $\Gamma$ into $\Delta$ given that the last ordinary genome is $\Pi$. The proof of Theorem 3 implies that $d(\Gamma, \Pi, \Delta)=n+|\Delta|+|\Gamma|-2H(\Pi)$.

Let $\Gamma'$ be the last ordinary genome produced by the algorithm. Consider an optimal scenario $O$ with a maximum number of duplications and let $\tilde{\Gamma}$ be the last ordinary linear genome in $O$. Since $O$ is optimal, $H(\tilde{\Gamma})$ must be maximal. Hence, $\tilde{\Gamma}$ cannot contain adjacencies with $\eta < 0$. Moreover, it cannot contain adjacencies with $\eta=0$ as such adjacencies increase $|\tilde{\Gamma}|$ and thus decrease the number of duplications in $O$. Therefore, $\tilde{\Gamma} \subseteq \Gamma'$.

We now show that $\forall \alpha \in \Gamma' \backslash \tilde{\Gamma}$, $\eta(\alpha)=1$. Suppose by contradiction that there is an adjacency $\alpha \in \Gamma' \backslash \tilde{\Gamma}$ with $\eta(\alpha) > 1$ and let $\Pi=\tilde{\Gamma} \cup \{\alpha\}$. If $\Pi$ is a linear genome, $d(\Gamma, \Pi, \Delta) < d(\Gamma, \tilde{\Gamma}, \Delta)$, contradicting the optimality of $O$. Otherwise, $\Pi$ contains a circular chromosome and by Lemma 3, there is an adjacency $\beta \in \tilde{\Gamma}$ with $\eta(\beta)=1$ such that $\Pi \backslash \{\beta\}$ is a linear genome with $H(\Pi \backslash \{\beta\}) > H(\tilde{\Gamma})$, again contradicting the optimality of $O$. Thus, $|\Gamma' \backslash \tilde{\Gamma}|=|\Gamma'|-|\tilde{\Gamma}|=H(\Gamma')-H(\tilde{\Gamma})$.

$\Gamma'$ may contain circular chromosomes. By Lemma 3, $\Gamma''$ is produced by removing one adjacency with $\eta=1$ from each circular chromosome in $\Gamma'$. Hence, $|\Gamma' \backslash \Gamma''|=|\Gamma'|-|\Gamma''|=H(\Gamma\prime)-H(\Gamma'')$.

Since both $\tilde{\Gamma}$ and $\Gamma''$ are last ordinary genomes, in optimal SCJD scenarios, $H(\tilde{\Gamma})=H(\Gamma'')$. Thus, $|\Gamma'|-|\tilde{\Gamma}|=H(\Gamma')-H(\tilde{\Gamma})=H(\Gamma')-H(\Gamma'\prime)=|\Gamma'|-|\Gamma''|$, which implies that $|\tilde{\Gamma}|=|\Gamma''|$. ■

One can decrease the number of duplications in an optimal SCJD scenario by adding adjacencies with $\eta(\alpha)=0$ to $\Gamma''$. However, we need to make sure that the resulting genome is still linear. Consider the following example:

*Example 6:* $\Gamma=a, b, c$, $\Delta=abccba$. From Theorem 4, we have that $\Gamma'=\Gamma$ and so the SCJD distance is 8. The scenario produced by Algorithm 1 will first duplicate the three chromosomes of $\Gamma$ and then perform five joins to create $\Delta$. An alternative optimal sorting scenario is as follows:

$$\Gamma \underset{JJ}{\rightarrow} abc \underset{D}{\rightarrow} abc, abc \underset{CC}{\rightarrow} abc, a, b, c \underset{JJJ}{\rightarrow} \Delta$$

Here, since each adjacency $\alpha \in \Delta$ has $\eta(\alpha)=0$, we chose $\Gamma''=abc$ and obtained an optimal scenario with a single duplication. In contrast, if we add to $\Gamma''$ the adjacencies $\{b_h, c_t\}$ and $\{c_h, b_t\}$ (which also have $\eta=0$), we create a circular chromosome and an invalid SCJD sorting scenario.

To minimize the number of duplications, we must add to $\Gamma''$ a maximum set of adjacencies with $\eta=0$ such that the resulting genome is still linear. Here, we show that this problem is NP-hard using a reduction similar to (Kováč, 2014).

**Theorem 6.** *Given an ordinary linear genome, $\Gamma$, a duplicated linear genome, $\Delta$, on the same gene set, and an integer, k, the problem of finding an optimal SCJD scenario with at most k duplications is NP-hard.*

*Proof.* Call a directed graph in which all in- and out-degrees are 2 a *2-digraph*. Deciding if a 2-digraph contains a Hamiltonian cycle is NP-hard (Plesnik, 1979; Kováč, 2014). This implies that the following variant is also NP-hard: given a 2-digraph, $G$, with an edge, $(x, y)$, decide if there is a Hamiltonian path from $y$ to $x$ in $G$.

Let $G = (V, E)$ be a 2-digraph with an edge $(x, y)$ as above. We may assume w.l.o.g. that $G$ is strongly connected since otherwise it would not contain a Hamiltonian path from $y$ to $x$. Notice that $G\backslash(x, y)$ contains an Eulerian path from $y$ to $x$ (Cormen et al., 2001). Denote it by $P = e_1, e_2, \ldots, e_m$.

We construct a duplicated genome $\Sigma$ as follows: for each $e_q = (u, v) \in P$, add the adjacency, $\{u^i_h, v^j_t\}$, where $i = 2$ if there is an edge $e_l = (u, v')$ with $l < q$ and $i = 1$ otherwise. Similarly, $j = 2$ if there is an edge $e_m = (u', v)$ with $m < q$ and $j = 1$ otherwise. The result is a linear chromosome created by traversing $P$ and numbering the first occurrence of each vertex $v$ in $P$ as the gene copy $v^1$ and the second occurrence as $v^2$. Denote by $\overset{\rightsquigarrow}{P}$ the sequence of genes along the path $P$. In addition, we add two new genes $w, z$ and the adjacencies $\{w^1_h, y^1_t\}$, $\{x^2_h, z^1_t\}$. Thus, $\Sigma$ has three linear chromosomes: $w^1 y^1 \underset{\rightsquigarrow}{P} x^2 z^1$, $w^2$ and $z^2$. Let $\Pi = \{\{w_h, y_t\}, \{x_h, z_t\}\}$ be an ordinary genome with $n$ chromosomes over the same set of genes. (Note that every vertex in $V\backslash\{x, y\}$ corresponds to a separate chromosome in $\Pi$).

Let $\Sigma_{(i)}$ and $\Pi_{(i)}$ be genomes in which every gene $v \in V$ is renamed $v_{(i)}$. We define $\Delta = \bigcup_{i=1}^{k} \Sigma_{(i)}$ and $\Gamma = \bigcup_{i=1}^{k} \Pi_{(i)}$ to be the disjoint union of $k$ different copies of $\Sigma$ and $\Pi$, respectively. This completes the reduction, which is clearly polynomial. We will show that there is an optimal SCJD scenario between $\Gamma$ and $\Delta$ with at most $k$ duplications if $G$ admits a Hamiltonian path from $y$ to $x$.

For each edge $e = (u, v) \in E$ and every $i$, the corresponding adjacency $\alpha = \{(u_{(i)})^j_h, (v_{(i)})^l_t\}$ has $\eta(\alpha) = 1$ if there are two parallel edges from $u$ to $v$ and $\eta(\alpha) = 0$ otherwise. In addition, for every $i$, $\eta(\{(w_{(i)})_h, (y_{(i)})_t\}) = \eta(\{(x_{(i)})_h, (z_{(i)})_t\}) = 1$, and every other adjacency of $w_{(i)}, z_{(i)}$ has $\eta < 0$.

Suppose $G$ contains a Hamiltonian path $S$ from $y$ to $x$. Let $\Gamma'$ be the genome formed by the set of adjacencies $\{\{(w_{(i)})_h, (y_{(i)})_t\}, \{(x_{(i)})_h, (z_{(i)})_t\} | i = 1 \ldots k\} \cup \{\{(u_{(i)})_h, (v_{(i)})_t\} | (u, v) \in S, i = 1 \ldots k\}$. Since $S$ is a Hamiltonian path, $\Gamma'$ is a valid ordinary linear genome with $k$ chromosomes of the form $w_{(i)} y_{(i)} \underset{\rightsquigarrow}{S} x_{(i)} z_{(i)}$. To prove that $\Gamma'$ maximizes $H(\cdot)$, we need to show it contains every adjacency with $\eta = 1$ and no adjacency with $\eta < 0$. Indeed (suppressing the copy index $i$ for clarity), the only adjacencies $\alpha$ with $\eta(\alpha) = 1$ are $\{w_h, y_t\}$, $\{x_h, z_t\}$ ($|\Delta_\alpha| = |\Gamma_\alpha| = 1$) and parallel edges in $G$ ($|\Delta_\alpha| = 2$, $|\Gamma_\alpha| = 0$), one copy of which must be included in $S$. All other adjacencies in $\Gamma'$ have $|\Delta_\alpha| = 1$, $|\Gamma_\alpha| = 0$ and $\eta(\alpha) = 0$. We conclude that $\Gamma'$ is part of an optimal scenario with $k$ duplications.

Conversely, suppose there is an optimal scenario $O^*$ with at most $k$ duplications and let $\tilde{\Gamma}$ be the last ordinary genome in $O^*$. Let $\Gamma' = \{\alpha | \eta(\alpha) > 0\}$ be a genome that minimizes the SCJD distance according to Theorem 4. First, notice that $\Gamma'$ is indeed a linear genome. Otherwise, a circular chromosome of adjacencies with $\eta(\alpha) = 1$ would imply a strongly connected component without the vertices $x, y$, contradicting the strong connectivity of $G$. It follows that $\Gamma' \subseteq \tilde{\Gamma}$, $H(\Gamma') = H(\tilde{\Gamma})$, and $\#_c \tilde{\Gamma} \leq k$.

Since $\Sigma_{(i)}$ and $\Sigma_{(j)}$ for $i \neq j$ contain different genes, an adjacency between a gene in $\Sigma_{(i)}$ and a gene $\Sigma_{(j)}$ has negative $\eta$. Therefore, $\tilde{\Gamma}$ contains no such adjacencies. Since $\tilde{\Gamma}$ has at most $k$ linear chromosomes, it must contain exactly $k$ linear chromosomes, each containing all the genes of $\Sigma_{(i)}$ for one $i$.

Let $C = w_{(1)} y_{(1)} \ldots x_{(1)} z_{(1)}$ be the linear chromosome in $\tilde{\Gamma}$ that contains all the genes of $\Sigma_{(1)}$. Define an edge set $S$ in $G$ by taking for each adjacency $\{(u_{(1)})_h, (v_{(1)})_t\} \in C \backslash \{\{(w_{(1)})_h, (y_{(1)})_t\}, \{(x_{(1)})_h, (z_{(1)})_t\}\}$ the edge $(u, v)$. Since $C$ is an ordinary linear chromosome containing all the genes of $\Sigma_{(1)}$, $S$ is a Hamiltonian path in $G$ from $y$ to $x$. ■

## 6. DISCUSSION

In this article, we presented the SCJD rearrangement model, which allows the operations, cut, join, and whole chromosome duplication. We analyzed the problem of finding the minimum number of SCJD operations that transform an ordinary linear genome into a duplicated linear genome and provided a linear time algorithm for it. Furthermore, we showed that this algorithm gives an optimal scenario with a maximum number of duplications and that finding one with fewest duplications is NP-hard.

In the analysis, we focused on the SCJD sorting problem, which restricts the target genome to have exactly two copies of each gene. However, it is not difficult to generalize our algorithm to address the more general situation where each gene in the target genome has *at most* two copies. One can show that in this case too, an optimal solution in which all duplications are consecutive exists. In addition, each adjacency in the original genome between a gene that has two copies and a gene that has one copy in the target genome must first be cut. This is true because duplications are defined over linear chromosomes in which every gene is unduplicated.

Our algorithm relies on the property that all duplications in the optimal solution can be clustered (Corollary 2). In this sense, the problem we study is similar to the SCJ Guided Genome Halving problem (Feijão and Meidanis, 2011). In that model, the whole genome is duplicated at once, while in ours, there is one duplication per chromosome, and accounting for these duplications is part of the optimization challenge.

Many aspects in the analysis of the SCJD model require further research: How can we address the problem if there are more than two copies of each gene? Can we find the SCJD distance between two arbitrary genomes—each containing single copy and multiple copy genes? How does removing the requirement of linearity affect various SCJD problems? Moreover, duplications may be defined differently, for example, tandem duplications (Bader, 2009) and segmental duplications (Shao et al., 2013). Finally, developing a rigorous model that will allow both duplications and deletions is needed to analyze the full complexity of real biological data such as cancer samples.

## ACKNOWLEDGMENTS

This study was supported by the Israeli Science Foundation (Grant No. 317/13) and the Dotan Hemato-Oncology Research Center at Tel Aviv University. R.Z. was supported, in part, by fellowships from the Edmond J. Safra Center for Bioinformatics at Tel Aviv University and from the Israeli Center of Research Excellence (I-CORE) Gene Regulation in Complex Human Disease (Center No 41/11).

## AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

## REFERENCES

Bader, M. 2009. Sorting by reversals, block interchanges, tandem duplications, and deletions. *BMC Bioinformatics* 10 (Suppl 1), S9.

Bader, M. 2010. Genome rearrangements with duplications. *BMC Bioinformatics* 11 (Suppl 1), S27.

Bayani, J., Selvarajah, S., Maire, G., et al. 2007. Genomic mechanisms and measurement of structural and numerical instability in cancer cells. *Semin. Cancer Biol.* 170, 5–18.

Bergeron, A., Mixtacki, J., and Stoye, J. 2006. A unifying view of genome rearrangements, 163–173. *In* Bücher, P., and Moret, B.M.E., eds. *Algorithms in Bioinformatics*, volume 4175 of *Lecture Notes in Computer Science*. Springer: Berlin, Heidelberg.

Biller, P., Feijão, P., and Meidanis, J. 2013. Rearrangement-based phylogeny using the Single-Cut-or-Join operation. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 100, 122–134.

Blanc, G., Barakat, A., Guyot, R., et al. 2000. Extensive duplication and reshuffling in the Arabidopsis genome. *Plant Cell* 120, 1093–1101.

Bulteau, L., Fertin, G., and Rusu, I. 2012. Sorting by transpositions is difficult. *SIAM J. Discrete Math.* 26, 1148–1180.

Caprara, A. 1997. Sorting by reversals is difficult, 75–83. *In Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB)*. ACM: New York, New York.

Christie, D.A. 1996. Sorting permutations by block-interchanges. *Inform. Process. Lett.* 60, 165–169.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., et al. 2001. *Introduction to Algorithms*, volume 2. MIT press, Cambridge.

Dias, Z., and Meidanis, J. 2001. Genome rearrangements distance by fusion, fission, and transposition is easy, 250. *In International Symposium on String Processing and Information Retrieval*. IEEE Computer Society.

Feijão, P., and Meidanis, J. 2011. SCJ: A breakpoint-like distance that simplifies several rearrangement problems. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 8, 1318–1329.

Fertin, G., Labarre, A., Rusu, I., et al. 2009. *Combinatorics of Genome Rearrangements*. MIT Press: Cambridge, MA.

Gordon, D.J., Resio, B., and Pellman, D. 2012. Causes and consequences of aneuploidy in cancer. *Nat. Rev. Genet.* 13, 189.

Giam, M., and Rancati, G. 2015. Aneuploidy and chromosomal instability in cancer: A jackpot to chaos. *Cell Div.* 10, 3.

Hannenhalli, S. 1996. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Appl. Math.* 71, 137–151.

Hannenhalli, S., and Pevzner, P.A. 1995. Transforming cabbage into turnip, 178–189. *In Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, volume 46. New York, New York.

Kováč, J. 2014. On the complexity of rearrangement problems under the breakpoint distance. *J. Comput. Biol.* 21, 1–15.

Lu, C.L., Huang, Y.L., Wang, T.C., et al. 2006. Analysis of circular genome rearrangement by fusions, fissions and block-interchanges. *BMC Bioinformatics* 7, 295.

Mira, C.V.G., and Meidanis, J. 2007. Sorting by block-interchanges and signed reversals. *ITNG* 7, 670–676.

Ozery-Flato, M., and Shamir, R. 2009. Sorting cancer karyotypes by elementary operations. *J. Comput. Biol.* 16, 1445–1460.

Plesnik, J. 1979. The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Inform. Process. Lett.* 8, 199–201.

Savard, O.T., Gagnon, Y., Bertrand, D., et al. 2011. Genome halving and double distance with losses. *J. Comput. Biol.* 18, 1185–1199.

Shao, M., and Lin, Y. 2012. Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. *BMC Bioinformatics* 13 (Suppl 19), S13.

Shao, M., Lin, Y., and Moret, B. 2013. Sorting genomes with rearrangements and segmental duplications through trajectory graphs. *BMC Bioinformatics* 14 (Suppl 15), S9.

Shao, M., Lin, Y., and Moret, B. 2014. An exact algorithm to compute the DCJ distance for genomes with duplicate genes, 280–292. *In* Sharan, R., ed. *Research in Computational Molecular Biology*, volume 8394 of *Lecture Notes in Computer Science*. Springer: Berlin, Heidelberg.

Tannier, E., Zheng, C., and Sankoff, D. 2009. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics* 10, 120.

Warren, R., and Sankoff, D. 2011. Genome aliquoting revisited. *J. Comput. Biol.* 18, 1065–1075.

Yancopoulos, S., Attie, O., and Friedberg, R. 2005. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340–3346.

Zeira, R., and Shamir, R. 2015. Sorting by cuts, joins and whole chromosome duplications, 396–409. *In Proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching*. Springer.

Address correspondence to:
*Mr. Ron Zeira*
*Tel Aviv University*
*Schreiber 011*
*Tel-Aviv 69978*
*Israel*

*E-mail:* ronzeira@post.tau.ac.il

# Chapter 3

# A Linear-Time Algorithm for the Copy Number Transformation Problem

JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 24, Number 12, 2017
© Mary Ann Liebert, Inc.
Pp. 1179–1194
DOI: 10.1089/cmb.2017.0060

# A Linear-Time Algorithm for the Copy Number Transformation Problem

RON ZEIRA,[1,*] MEIRAV ZEHAVI,[2,*] and RON SHAMIR[1]

## ABSTRACT

**Problems of genome rearrangement are central in both evolution and cancer. Most evolutionary scenarios have been studied under the assumption that the genome contains a single copy of each gene. In contrast, tumor genomes undergo deletions and duplications, and thus, the number of copies of genes varies. The number of copies of each segment along a chromosome is called its copy number profile (CNP). Understanding CNP changes can assist in predicting disease progression and treatment. To date, questions related to distances between CNPs gained little scientific attention. Here we focus on the following fundamental problem, introduced by Schwarz et al.: given two CNPs, *u* and *v*, compute the minimum number of operations transforming *u* into *v*, where the edit operations are segmental deletions and amplifications. We establish the computational complexity of this problem, showing that it is solvable in linear time and constant space.**

**Keywords:** copy number, edit distance, genome rearrangement.

## 1. INTRODUCTION

The genome of a species evolves by undergoing small and large mutations over generations. Large mutations modify genome organization by rearrangement of genomic segments. Computational analysis of the process of *genome rearrangement* has been the subject of extensive research over the last two decades (Fertin et al., 2009). The majority of these studies to date were restricted to a single copy of each gene and were concerned with the reordering of segments. Extant models that do not make this assumption often result in NP-hard problems (Tannier et al., 2009; Savard et al., 2011; Shao and Lin, 2012).

While most work on genome rearrangements to date was done in the context of species evolution, there is today great opportunity in analysis of cancer genome evolution. Cancer is a dynamic process characterized by the rapid accumulation of somatic mutations, which produce complex tumor genomes. Species evolution happens over eons and changes are carried over from one generation to the next. In contrast, cancer evolution happens within a single individual over a few decades. In many tumor genomes, a lot of the changes are segmental deletions and amplifications (The Cancer Genome Atlas Research Network, 2011). As a result, the number of copies of each segment along a chromosome, known as its copy number profile (CNP), changes during cancer development, compared to the normal genome that has two copies

[1]Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel.
[2]Department of Informatics, University of Bergen, Bergen, Norway.
*These authors contributed equally to this work.

(or *alleles*) for each segment. Understanding these changes can assist in predicting disease progression and the outcome of medical interventions. However, computational questions related to distances between CNPs received little scientific attention to date. Such questions are the topic of this article.

Over the years, a variety of methods were used to determine the CNP of a cancer genome, at different resolutions. G-banding allows viewing the chromosome bands (Pinkel et al., 1986). Fluoroscent In Situ Hybridization (FISH) measures the copy numbers of tens to hundreds of targeted genes (Chowdhury et al., 2014). Array comparative genomic hybridization gives a higher resolution of CN estimation for a cell population (Urban et al., 2006). Most recently, deep sequencing techniques yield CNPs by using read depth data (Oesper et al., 2012). While it would have been preferable to analyze the genome (karyotype) itself and not its CNP, detection of structural variations from sequencing data is still problematic (McPherson et al., 2012; Abo et al., 2014). Today it is a routine procedure to obtain detailed CNPs of cancer genomes, but utilizing them to understand cancer evolution is still an open problem.

Given two CNPs, the healthy tissue's and the tumor's, evaluating the distance between them can help in understanding cancer progression. A naive measure of distance is the Euclidean distance between the two profiles (Schwarz et al., 2014). Chowdhury et al. defined edit distance between CNPs obtained from FISH, where the edit operations are amplification or deletion of single genes, single chromosomes, or the whole genome (Chowdhury et al., 2013, 2014, 2015). However, calculating these distances requires exponential time in the number of genes and therefore is limited to low-resolution FISH data. The *TuMult* algorithm uses the number of breakpoints (loci where the CNs change) between two profiles as a simple distance measure (Letouzé et al., 2010).

Schwartz et al. introduced a model that admits amplification and deletion of contiguous segments (Schwarz et al., 2014). The edit distance between two CNPs was defined as the minimum number of segmental deletions and duplications over all separations of the profiles into two alleles (a procedure known as *phasing*). Their algorithm *MEDICC* for computing the edit distance uses finite-state transducers (FSTs) (Mohri, 2003) to model the profiles and efficiently compute the distance. However, the complexity of this method was not analyzed. Even without the phasing computation, the method needs to compose a three-state transducer with itself $B$ times, resulting in a transducer with $3^B$ states (Mohri, 2004; Schwarz et al., 2014). Here, $B$ is the maximum CN in the input. The running time of FST procedures relies on the number of states and transitions, and in some cases may be exponential (Mohri, 2003, 2004).

### 1.1. *Copy number transformation*

We investigate the following problem, which underlies the model of Schwarz et al. (2014): Given two CNPs, $u$ and $v$, compute the minimum number of segmental duplications and deletions needed to transform $u$ into $v$. We call this problem the Copy Number Transformation Problem (CNTP). A CNP is represented by a vector of non-negative integers (the number of copies of each segment). A segmental deletion (amplification) decreases (resp. increases) by 1 the values of a contiguous interval of the vector, where zero values are not affected. Formal definitions are given in Section 2.

### 1.2. *Our contribution*

We show that the CNTP is solvable in linear time and constant space. The algorithm relies on several properties of the problem that we establish in Section 3.1, which may also be relevant to the analysis of other problems involving CNPs. By exploiting these properties, we obtain a pseudopolynomial dynamic programming algorithm for CNTP, presented in Section 3.2. In Section 3.3, by establishing that a certain function in the dynamic programming recursion is piecewise linear, we improve its performance and obtain our main result, namely, a linear-time algorithm for CNTP.

Preliminary version of this article appeared in the proceedings of CPM 2016 (Shamir et al., 2016).

## 2. PRELIMINARIES

In this section, we give definitions and notations that are used throughout the article. Let $n \in \mathbb{N}$. A CNP is a vector $V = (v_1, v_2, \ldots, v_n)$, where $v_i \in \mathbb{N} \cup \{0\}$. Each position in $V$ corresponds to a segment in the normal genome, where the segments are ordered as in the normal genome. For simplicity we call a position a *gene*. A CN operation (CNO) is a triple $c = (\ell, h, w)$, where $1 \leq \ell \leq h \leq n$ and $w \in \{-1, 1\}$. We say that

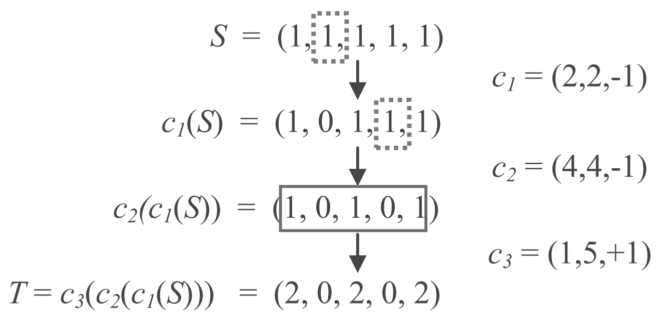
$S = (1, 1, 1, 1, 1)$
$c_1 = (2,2,-1)$
$c_1(S) = (1, 0, 1, 1, 1)$
$c_2 = (4,4,-1)$
$c_2(c_1(S)) = (1, 0, 1, 0, 1)$
$c_3 = (1,5,+1)$
$T = c_3(c_2(c_1(S))) = (2, 0, 2, 0, 2)$

**FIG. 1.** The CNT $C = (c_1, c_2, c_3)$ transforms $S$ into $T$. The size of $C$ is 3. Red (dotted) and green (solid) blocks indicate deletions and amplifications, respectively. CNT, CN transformation.

a CNO $c = (\ell, h, -1)$ is a *deletion* and $c = (\ell, h, 1)$ is an *amplification*. Given a CNP $V = (v_1, v_2, \ldots, v_n)$ and a CNO $c = (\ell, h, w)$, we define the operation $c(V) = (c(v_1), c(v_2), \ldots, c(v_n))$ as follows. For each $i \in \{1, 2, \ldots, n\}$, if $\ell \leq i \leq h$ and $v_i \geq 1$, then $c(v_i) = v_i + w$, otherwise (i.e., if $i < \ell$ or $i > h$ or $v_i = 0$) $c(v_i) = v_i$. A triple $c = (\ell, h, w)$ with $h < \ell$ has no effect on the CNP, that is, $c(V) = V$. Given two CNPs, $S = (s_1, s_2, \ldots, s_n)$ (source) and $T = (t_1, t_2, \ldots, t_n)$ (target), a CN transformation (CNT) is a vector $C = (c_1, c_2, \ldots, c_m)$, where $m \in \mathbb{N}$ and each $c_i = (\ell_i, h_i, w_i)$ is a CNO, such that $C(S) = c_m(c_{m-1}(\cdots(c_1(S)))) = T$. The *size* of $C$, denoted $|C|$, is $m$. An example is given in Figure 1. Finally, we denote the number of operations of weight $w \in \{-1, 1\}$ affecting $s_i$ by $op(C, w, i) = |\{(\ell, h, w) \in C : \ell \leq i \leq h\}|$. For example, in Figure 1, $op(C, -1, 2) = 1$.

The CN distance from $S$ to $T$, $\text{dist}(S, T)$, is the smallest size of a CNT $C$ that satisfies $C(S) = T$, where if no such CNT exists, $\text{dist}(S, T) = \infty$. Note that *dist* is not symmetric. For example, for $S = (1)$ and $T = (0)$, $\text{dist}(S, T) = 1$ but $\text{dist}(T, S) = \infty$. Given two CNPs, $S = (s_1, s_2, \ldots, s_n)$ and $T = (t_1, t_2, \ldots, t_n)$, the CNTP seeks $\text{dist}(S, T)$ (if one exists). We say that a CNT $C$ is *optimal* if it realizes $\text{dist}(S, T)$, that is, $|C| = \text{dist}(S, T)$ (there may exist several optimal CNTs). We let $B = \max\{\max_{i=1}^{n}\{s_i\}, \max_{i=1}^{n}\{t_i\}\}$ denote the maximum CN in the input. Finally, for all $1 \leq i \leq n$, we define $u_i = s_i - t_i$.

## 3. AN ALGORITHM FOR CNTP

We first present an $O(nB^2)$-time and $O(B)$-space algorithm for CNTP, based on dynamic programming (Sections 3.1 and 3.2). Recall that $B$ is the maximal integer in the input, so that algorithm is pseudopolynomial. Then, we modify this algorithm to run in linear time (Section 3.3). On a high level, the modification is based on the observation that the table used by the algorithm to store values of partial solutions can be described by $O(n)$ piecewise linear functions, where each function encapsulates $O(B)$ entries of the table. We show that each function has only three linear segments, and so, the computation of an entry can be performed in time $O(1)$ rather than $O(B)$. Furthermore, since each function can be represented in a compact manner, the size of table shrinks from $O(nB)$ to $O(n)$. The precise definitions of the table and the functions are given in Sections 3.2 and 3.3. Our proof of the correctness of the use of these functions requires a somewhat extensive case analysis that is presented separately in Section 3.4.

### 3.1. Key propositions

We start by developing `DpCntpAlg`, an $O(nB^2)$-time dynamic programming algorithm for CNTP. Let $(S = (s_1, s_2, \ldots, s_n),\ T = (t_1, t_2, \ldots, t_n))$ be the input. Observe that there exists a CNT $C$ such that $C(S) = T$ if and only if there does not exist an index $1 \leq i \leq n$ such that $s_i = 0$ and $t_i > 0$. Since the existence of such an index can be determined in linear time (where, if such an index is found, we return $\infty$), we will assume that $\text{dist}(S, T) < \infty$. To simplify the presentation, we further assume w.l.o.g. that $t_1, t_n \neq 0$. Indeed, if $t_1 = 0$ or $t_n = 0$, we can solve the input $(S' = (1, s_1, s_2, \ldots, s_n, 1), T' = (1, t_1, t_2, \ldots, t_n, 1))$ instead, since it holds that $\text{dist}(S, T) = \text{dist}(S', T')$. Finally, we assume w.l.o.g. for all $1 \leq i \leq n$, $s_i > 0$. Indeed, if there exists $1 \leq i \leq n$ such that $s_i = 0$, then also $t_i = 0$, and we can solve the input $(S' = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n), T' = (t_1 \ldots, t_{i-1}, t_{i+1}, \ldots, t_n))$ since $\text{dist}(S, T) = \text{dist}(S', T')$.

`DpCntpAlg` exploits four key observations about the nature of the problem at hand, summarized as follows: (1) it is sufficient to examine CNTs where all of the deletions precede all of the amplifications; (2)

it is sufficient to examine CNTs that do not contain both a deletion that affects $s_i$ but not $s_{i+1}$ and a deletion that affects $s_{i+1}$ but not $s_i$, and the same is true for amplifications; (3) when seeking an optimal solution, it is not necessary to store information indicating how many deletions/amplifications affect $s_i$ if $t_i = 0$; and (4) the maximum number of deletions/amplifications that affect each $s_i$ can be bounded by $B$.

To formally state the first observation, we need the following definition.

**Definition 1.** *A CNT $C = (c_1, c_2, \ldots, c_m)$ is ordered if for all $1 \le i < j \le m$, if $c_j$ is a deletion, then $c_i$ is also a deletion.*

**Proposition 1.** *There exists an optimal ordered CNT.*

We note that the ''opposite'' proposition, stating that there exists an optimal CNT where all of the amplifications precede all of the deletions, does not hold: consider, for example, $S = (1, 1, 1, 1, 1)$ and $T = (2, 0, 2, 0, 2)$. To prove this proposition, we will need the following claim.

**Claim 1.** *Let $C = (c_1, c_2, \ldots, c_m)$ be an optimal CNT and let i be an index such that $c_i = (\ell_i, h_i, 1)$ and $c_{i+1} = (\ell_{i+1}, h_{i+1}, -1)$. Then, there exists an optimal CNT $C' = (c_1, \ldots, c_{i-1}, c'_i, c'_{i+1}, c_{i+2}, \ldots, c_m)$, where $c'_i = (\ell'_i, h'_i, w'_i)$ and $c'_{i+1} = (\ell'_{i+1}, h'_{i+1}, w'_{i+1})$, such that one of the following conditions holds.*

1. $(h'_i - \ell'_i) + (h'_{i+1} - \ell'_{i+1}) < (h_i - \ell_i) + (h_{i+1} - \ell_{i+1})$.
2. $(h'_i - \ell'_i) + (h'_{i+1} - \ell'_{i+1}) = (h_i - \ell_i) + (h_{i+1} - \ell_{i+1})$ *and* $w'_i = -1$.

*Proof.* Consider the following exhaustive case analysis (Fig. 2). ■

I. $h_i < \ell_{i+1}$ or $h_{i+1} < \ell_i$: In this case, the segments corresponding to $c_i$ and $c_{i+1}$ are disjoint. Thus, we can simply define $c'_i = c_{i+1}$ and $c'_{i+1} = c_i$. Then, Condition 2 is satisfied.

II. $\ell_i \le \ell_{i+1} \le h_i \le h_{i+1}$: Define $c'_i = (h_i + 1, h_{i+1}, -1)$ and $c'_{i+1} = (\ell_i, \ell_{i+1} - 1, 1)$. For any CNP $V = (v_1, v_2, \ldots, v_n)$, $c'_{i+1}(c'_i(V)) = c_{i+1}(c_i(V))$. This argument holds because an application of $c_i$, followed by an application of $c_{i+1}$, does not change any entry $v_k$ such that $\ell_{i+1} \le k \le h_i$. We have that $C'(S) = T$. Since $|C'| = |C|$, $C'$ is an optimal CNT. Now, Condition 1 is satisfied.

III. $\ell_{i+1} \le \ell_i \le h_{i+1} \le h_i$: Define $c'_i = (\ell_{i+1}, \ell_i - 1, -1)$ and $c'_{i+1} = (h_{i+1} + 1, h_i, 1)$. As in the second case, we obtain an optimal CNT that satisfies Condition 1.

IV. $\ell_i \le \ell_{i+1} \le h_{i+1} \le h_i$: Define $c'_i = (\ell_i, \ell_{i+1} - 1, 1)$ and $c'_{i+1} = (h_{i+1} + 1, h_i, 1)$. As in the second case, we obtain an optimal CNT that satisfies Condition 1.

V. $\ell_{i+1} \le \ell_i \le h_i \le h_{i+1}$: Define $c'_i = (\ell_{i+1}, \ell_i - 1, -1)$ and $c'_{i+1} = (h_i + 1, h_{i+1}, -1)$. As in the second case, we obtain an optimal CNT that satisfies Condition 1.

As we show below, Claim 1 implies the existence of an ordered optimal CNT. In each of the cases in Claim 1, a local change is made in the CNT. Note, however, that just performing enough local operations does not guarantee reaching an ordered optimal CNT. For example, in a CNT with three consecutive CNOs, $c_i = (\ell_i, h_i, 1)$, $c_{i+1} = (\ell_{i+1}, h_{i+1}, 1)$, $c_{i+2} = (\ell_{i+2}, h_{i+2}, -1)$, one may loop between changing $c_{i+1}$ into a deletion and then into an amplification.

*Proof (Proof of Proposition 1).* Let $\mathcal{C}$ be the set of optimal CNTs, and suppose, by way of contradiction, that it does not contain an ordered CNT. The three following phases sieve some solutions out of $\mathcal{C}$. Informally, we initially consider only optimal CNTs that minimize the sum of the sizes of the segments corresponding to their CNOs ($\mathcal{C}^1$); then, we further consider only the CNTs whose first amplification is as late as possible ($\mathcal{C}^2$); finally, we only take the CNTs whose first deletion after their first amplification is as early as possible ($\mathcal{C}^3$). An illustration is given in Figure 3.

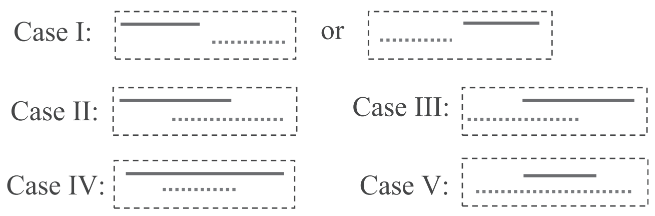

**FIG. 2.** The proof of Claim 1. The green (solid) lines correspond to $c_i$, and the red (dotted) lines correspond to $c_{i+1}$.

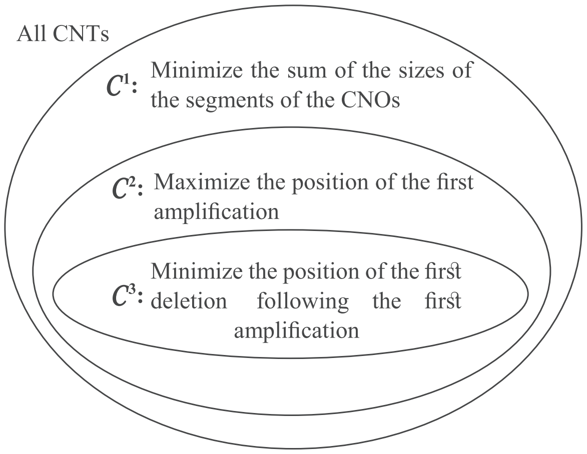
All CNTs
$\mathcal{C}^1$: Minimize the sum of the sizes of the segments of the CNOs
$\mathcal{C}^2$: Maximize the position of the first amplification
$\mathcal{C}^3$: Minimize the position of the first deletion following the first amplification

**FIG. 3.** The proof of Proposition 1.

- Given $C=(c_1, c_2, \ldots, c_m) \in \mathcal{C}$, define $x(C)=\sum_{i=1}^{m}(h_i-\ell_i)$. Let $\mathcal{C}^1$ be the set of every $C \in \mathcal{C}$ for which there does not exist $C' \in \mathcal{C}$ such that $x(C) > x(C')$.
- Given $C=(c_1, c_2, \ldots, c_m) \in \mathcal{C}^1$, let $y(C)$ be the largest index $0 \leq i \leq m$ such that for all $1 \leq j \leq i$, $c_j$ is a deletion. Note that $y(C)=0$ if and only if $c_1$ is an amplification. Let $\mathcal{C}^2$ be the set of every $C \in \mathcal{C}^1$ for which there does not exist $C' \in \mathcal{C}^1$ such that $y(C) < y(C')$.
- Given $C=(c_1, c_2, \ldots, c_m) \in \mathcal{C}^2$, let $z(C)$ be the smallest index $i \in \{y(C)+1, \ldots, m\}$ such that $c_i$ is a deletion. By the definition of $y(C)$ and since $C$ is not ordered, we have that $z(C)$ is well defined and $z(C) \geq y(C)+2$. Let $\mathcal{C}^3$ be the set of every $C \in \mathcal{C}^2$ for which there does not exist $C' \in \mathcal{C}^2$ such that $z(C) > z(C')$.

Since $\mathcal{C} \neq \emptyset$, we have that $\mathcal{C}^3 \neq \emptyset$. Thus, we can let $C=(c_1, c_2, \ldots, c_m)$ be a solution in $\mathcal{C}^3$. Let $i$ be the smallest index such that $c_i$ is an amplification and $c_{i+1}$ is a deletion. Now, consider the conditions in Claim 1: if Condition 1 holds, we have a contradiction to the fact that $C \in \mathcal{C}^1$, while if Condition 2 holds, we have a contradiction either to the fact that $C \in \mathcal{C}^2$ (if $i=1$ or $c_{i-1}$ is a deletion) or to the fact that $C \in \mathcal{C}^3$ (otherwise). Thus, we conclude that $\mathcal{C}$ contains an ordered CNT.

**Definition 2.** *A CNT C is elongated if for all* $1 \leq i < n$ *and* $w \in \{-1, 1\}$,

$$\min\{op(C,w,i), op(C,w,i+1)\} = |\{(\ell,h,w) \in C : \ell \leq i, i+1 \leq h\}|.$$

Equivalently, $C$ is elongated if no two amplifications (or deletions) ‘‘dovetail,’’ that is, one ending at $i$ and the other starting at $i+1$. It is clear that for any CNT $C$, the inequality $\geq$ holds above (since $\{(\ell,h,w) \in C : \ell \leq i, i+1 \leq h\}$ is a subset of both $\{(\ell,h,w) \in C : \ell \leq i \leq h\}$ and $\{(\ell,h,w) \in C : \ell \leq i+1 \leq h\}$). Our second key proposition implies the inequality $\leq$ holds as well. An example for an elongated CNT is given in Figure 4A.

To prove Proposition 2, we will need the following claim.

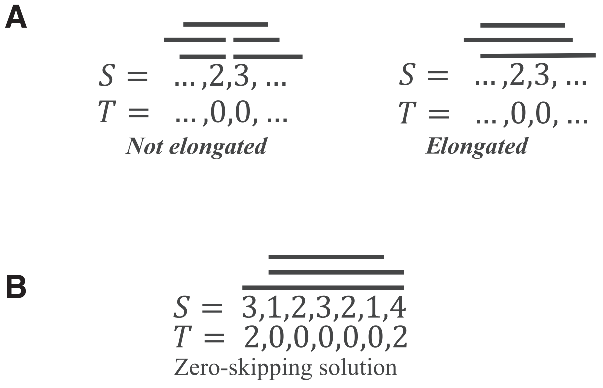

**FIG. 4.** **(A)** Elongated and nonelongated CNTs. **(B)** A zero-skipping solution. The top lines indicate the range of deletions.

**Claim 2.** *Let $C=(c_1, c_2, \ldots, c_m)$ be an optimal ordered CNT, and let $1 \leq i < j \leq m$ be indices such that either both $c_i$ and $c_j$ are deletions or both $c_i$ and $c_j$ are amplifications. Then, the CNT $C'$ obtained from $C$ by swapping the locations of $c_i$ and $c_j$ is also an optimal ordered CNT.*

*Proof.* Clearly, $C'$ is ordered and $|C'|=|C|$. Thus, it is sufficient to show that $C'(S)=C(S)$. Observe that because $C$ is ordered, for any $1 \leq q \leq n$, the value of the $q^{st}$ CN in $C(S)$ is $x+y$, where $x=\max\{s_q - op(C, -1, q), 0\}$, $y=0$ if $x=0$, and $y=op(C, 1, q)$ otherwise. By the definition of $C'$ (which is also ordered and contains the same CNOs as $C$), this is also the value of the $q^{st}$ CN in $C'(S)$. ∎

We are now ready to show the following property.

**Proposition 2.** *Every ordered optimal CNT is elongated.*

*Proof.* Let $C=(c_1, c_2, \ldots, c_m)$ be an optimal ordered CNT. Suppose that, by way of contradiction, $C$ is not elongated. Thus, there exist $1 \leq i < n$ and $w \in \{-1, 1\}$ such that

$$\min\{op(C, w, i), op(C, w, i+1)\} > |\{(\ell, h, w) \in C : \ell \leq i, i+1 \leq h\}|.$$

Therefore, $C$ contains two CNOs $c_p=(\ell_p, h_p, w)$ and $c_q=(\ell_q, h_q, w)$ such that $h_p=i$ and $\ell_q=i+1$. By Claim 2, we can assume that $p=q+1$. Now, by removing $c_p$ and replacing $c_q$ by the CNO $c=(\ell_p, h_q, w)$, we obtain a CNT $C'$ such that $C'(S)=T$. However, $|C'| < |C|$, which contradicts the optimality of $C$. ∎

To formalize our third key proposition, we need the following definition.

**Definition 3.** *A CNT $C$ is zero-skipping if for every $1 \leq i < j \leq n$ such that for all $i < r \leq j$, $t_r=0$ we have*

$$op(C, -1, j)=\max\left\{\max_{r=i+1}^{j}\{s_r\}, op(C, -1, i)\right\}, \text{ and } op(C, 1, j)=op(C, 1, i).$$

In words, for a block of consecutive zeros in the target profile, all deletions that span the block also include its flanking positions. An example of a zero-skipping CNT is given in Figure 4B.

**Proposition 3.** *There exists an optimal ordered zero-skipping CNT.*

*Proof.* By Proposition 1, there is an optimal ordered CNT $C=(c_1, c_2, \ldots, c_m)$. If $C$ is zero-skipping, we are done, and thus we next suppose that it does not. Thus, there exists $1 \leq i < j \leq n$ such that $t_r=0$ for all $i < r \leq j$, for which at least one of the following conditions is satisfied. ∎

1. $op(C, -1, j) \neq \max\{\max_{r=i+1}^{j}\{s_r\}, op(C, -1, i)\}$.
2. $op(C, 1, j) \neq op(C, 1, i)$.

We can assume w.l.o.g. that $j$ is the smallest index that is larger than $i$ for which at least one of the above conditions is satisfied. Thus, at least one of the following conditions is satisfied.

1. $op(C, -1, j) \neq \max\{s_j, op(C, -1, j-1)\}$.
2. $op(C, 1, j) \neq op(C, 1, j-1)$.

Since $t_j=0$, $|\{(\ell, h, -1) \in C : \ell \leq j \leq h\}| \geq s_j$. Moreover, because $C$ is ordered and $t_j=0$, we can replace each CNO $c=(\ell, h, w)$ in $C$ such that $h=j-1$ by the CNO $c'=(\ell, j, w)$. Thus, we overall obtain an optimal ordered CNT $C'$, such that, if it is not zero-skipping (in which case we are done), at least one of the following conditions is satisfied.

1. $op(C, -1, j) > \max\{s_j, op(C, -1, j-1)\}$.
2. $op(C, 1, j) > op(C, 1, j-1)$.

Since $C'$ is ordered and $t_j=0$, we can choose a CNO $c=(\ell, h, w)$ in $C'$ such that $\ell=j$, as well as $w=-1$ if the first condition is satisfied and $w=1$ otherwise, and replace it by the CNO $c'=(j+1, h, w)$. This operation results in an optimal ordered CNT. By repeating it enough times, we obtain an optimal ordered CNT that is zero-skipping. ∎

For a position with positive target value, knowing the number of deletions that affected it uniquely determines the number of amplifications that affected it. This simple fact will help the efficiency of our procedures. Formally:

**Observation 1.** *Let $1 \leq i \leq n$ be an index such that $t_i > 0$, and let $C=(c_1, c_2, \ldots, c_m)$ be a CNT such that $C(S)=T$. Then, $op(C, 1, i) = -u_i + op(C, -1, i)$.*

Finally, we formalize our fourth key proposition.

**Definition 4.** *A CNT $C$ is bounded if for all $1 \leq i \leq n$ and every $w \in \{-1, 1\}$, we have $op(C, w, i) \leq B$.*

**Proposition 4.** *Every optimal ordered CNT that is zero-skipping is also bounded.*

*Proof.* Let $C$ be an optimal ordered CNT that is zero-skipping. Suppose, by way of contradiction, that $C$ is not bounded. That is, there exists $1 \leq i \leq n$ and $w \in \{-1, 1\}$ such that $op(C, w, i) > B$. First suppose that $t_i > 0$. Then, since $C$ is ordered and $C(S)=T$, we have that $w=1$. However, this contradicts the correctness of Observation 1. Thus, we can next suppose that $t_i = 0$, which also implies that $i > 1$. We also assume w.l.o.g. that $i$ is the smallest index such that $op(C, w, i) > B$. Therefore, at least one of the following conditions is satisfied.

1. $op(C, -1, j) > \max\{s_j, op(C, -1, j-1)\}$.
2. $op(C, 1, j) > op(C, 1, j-1)$.

Thus, we necessarily obtain a contradiction to the fact that $C$ is zero-skipping. ■

## 3.2. *An* O(nB$^2$)*-time algorithm for CNTP*

On a high-level, the dynamic programming algorithm works as follows. It considers increasing prefixes $S^i = (s_1, s_2, \ldots, s_i)$ and $T^i = (t_1, t_2, \ldots, t_i)$ of the input. It computes a table M having $n(B+1)$ entries where M$[i, d]$ is the best value of a solution on $(S^i, T^i)$ that uses exactly $d$ deletions that affect the $i^{\text{th}}$ position. The parameter $d$ ranges between zero and $B$, and the values for each $i$ are computed based on values M$[j, \cdot\,]$ for a single specific $j < i$. In particular, at each point of time, only two rows of the table M are stored. By Propositions 1–4, the algorithm considers only ordered, elongated, zero-skipping and bounded solutions. We call such solutions *good*.

More formally, given $1 \leq i \leq n$ and $0 \leq d \leq B$, we say that a CNT $C$ is an *$(i, d)$-CNT* if $C(S^i) = T^i$, $d = op(C, -1, i)$, and $C$ is good. We say that an $(i, d)$-CNT $C$ is *optimal* if there is no $(i, d)$-CNT $C'$ such that $|C'| < |C|$. Our goal will be to ensure that each entry M$[i, d]$ stores the size of an optimal $(i, d)$-CNT, where if no such CNT exists, it stores $\infty$. We do not compute entries M$[i, d]$ such that $t_i = 0$; indeed, by relying on Property 3, we are able to skip such entries (although our recursive formula does consider CNs $s_i$ referring to indices $i$ such that $t_i = 0$). In this context, observe that any ordered CNT $C$ such that $C(S) = T$ consists of at least $u_i$ deletions that affect $s_i$, and if $t_i > 0$, it cannot consist of more than $s_i - 1$ such deletions (since after decreasing $s_i$ to 0, it remains 0). Moreover, if $u_i \leq d < s_i$, there exists an $(i, d)$-CNT—by independently adjusting the value of each position $< i$ to its target position and the value at position $i$ with $d$ deletions, using operations of span 1.

**Observation 2.** *Given $1 \leq i \leq n$ such that $t_i > 0$ and $0 \leq d \leq B$, there exists an $(i, d)$-CNT if and only if $u_i \leq d < s_i$.*

In case $s_i < t_i$, Observation 2 states that there exists an $(i, d)$-CNT if and only if $d < s_i$. In light of this observation, we will use the following assumption.

**Assumption 1.** *In the computation below, we assume that* $\max\{u_i, 0\} \leq d < s_i$. *Entries $M[i, d]$ for which it is not true that* $\max\{u_i, 0\} \leq d < s_i$ *store $\infty$.*

By Observation 1, if a solution involved $d$ deletions at position $i$ with $t_i > 0$, then it involved $-u_i + d$ amplifications at that position. For convenience denote that number by $a(i, d) = -u_i + d$ for all $1 \leq i \leq n$ satisfying $t_i > 0$ and $\max\{u_i, 0\} \leq d < s_i$, and $a(i, d) = \infty$ otherwise.

For input profiles $S$, $T$, the algorithm precomputes two vectors. Given an index $1 < i \leq n$ such that $t_i > 0$, let prev$(i)$ denote the largest index $j < i$ such that $t_j > 0$. Moreover, if prev$(i) = i - 1$, let $Q_i = 0$, and otherwise let $Q_i = \max_{\text{prev}(i) < j < i}\{s_j\}$. A zero-skipping solution (Fig. 5) will skip the positions between $i$ and prev$(i)$ in the computation, but will make sure to perform at least $Q_i$ deletions spanning the skipped positions.

**Initialization:** The initialization step sets all entries M$[1, d]$ as follows.

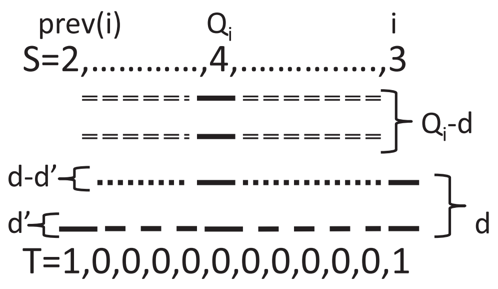

**FIG. 5.** Zero-skipping in the recursive formula. $T$ has a maximal block of zeros between positions prev($i$) and $i$, $S$ has values 2 and 3, respectively, in these positions and a maximum value 4 within the interval of genes, attained at position $Q_i$. $d'$ deletions can be elongated from prev($i$) up to position $i$. $d-d'$ deletions can be extended forward up to position $i$ and backward to position prev($i$)+1. In addition, $Q_i-d$ additional deletions are needed to delete $Q_i$.

$$\mathrm{M}[1, d] \leftarrow d + a(1, d).$$

**Recursion:** If $t_i = 0$ position $i$ is skipped. Suppose that $i > 1$, $t_i > 0$, and $\max\{u_i, 0\} \leq d < s_i$. The order of the computation is determined by the first argument. The computation is summarized in the following formula and illustrated in Figure 5.

$$\begin{aligned}\mathrm{M}[i, d] \leftarrow \min_{0 \leq d' \leq B} \{&\mathrm{M}[\mathrm{prev}(i), d'] + \max\{d-d', 0\} \\ &+ \max\{a(i, d) - a(\mathrm{prev}(i), d'), 0\} + \max\{Q_i - \max\{d, d'\}, 0\}\}\end{aligned} \tag{1}$$

Roughly speaking, to compute $M[i, d]$ we look back to the previous nonzero position in $T$, and for each value $d'$ in that position add the difference from $d$ if needed, the number of amplifications to be added if needed, and the number of additional deletions if such are needed to take care of the zero positions that were skipped. After filling the table M, `DpCntpAlg` returns $\min_{0 \leq d \leq B} \mathrm{M}[n, d]$. The full algorithm is given in Algorithm 1. An example of a partially filled table is given in Figure 6.

**Algorithm 1:** `DpCntpAlg`

```
Input: $S$, $T$, $Q$, $prev$
Output: $dist(S, T)$
  for $d = 1, \ldots, B$ do
    $M[1, d] \leftarrow d + a(1, d)$
  end for
  for $i = 2, \ldots, n$, $t_i > 0$ do
    for $d = 0, \ldots, B$ do
      if $\max\{u_i, 0\} \leq d < s_i$ then
        $\mathrm{M}[i,d] \leftarrow \min_{0 \leq d' \leq B}\{\mathrm{M}[\mathrm{prev}(i), d'] + \max\{d - d', 0\} + \max\{a(i, d) - a(\mathrm{prev}(i), d'), 0\}$
      else
        $\mathrm{M}[i, d] \leftarrow \infty$
      end if
    end for
  end for
  return $\min_{0 \leq d \leq B} \mathrm{M}[n, d]$
```

$M[i, d]$

| | i = 1 | i = 7 |
|---|---|---|
| d = 0 | ∞ | ∞ |
| d = 1 | 1 | ∞ |
| d = 2 | 2 | 3 |
| d = 3 | ∞ | 4 |
| d = 4 | ∞ | ∞ |

**FIG. 6.** The DP $M[i, d]$ matrix for the two CNPs in Figure 4B.

**Correctness:** First, we claim that the entries of the table M are computed properly.

**Lemma 1.** *For all* $1 \leq i \leq n$ *such that* $t_i > 0$ *and for all* $0 \leq d \leq B$, $M[i, d]$ *stores the size of an optimal* $(i, d)$*-CNT, where if no such CNT exists, it stores* $\infty$.

*Proof.* We prove the lemma by induction on the order of the computation. ■

The correctness of the initialization step follows from the definition of an $(i, d)$-CNT and Observation 1. Now, fix $1 < i \leq n$ such that $t_i > 0$, and fix $\max\{u_i, 0\} \leq d < s_i$. Let $m$ be the size of an optimal $(i, d)$-CNT. Suppose that the lemma is correct for all $i' < i$ and $0 \leq d' \leq B$. We need to show that $\mathrm{M}[i, d] = m$.

**First Direction:** First, we show that $\mathrm{M}[i, d] \leq m$. Let $C = (c_1, c_2, \ldots, c_m)$ be an optimal $(i, d)$-CNT, and for all $1 \leq j \leq m$, denote $c_j = (\ell_j, h_j, w_j)$. For all $1 \leq j \leq m$, let $c_j' = (\ell_j, \min\{h_j, \mathrm{prev}(i)\}, w_j)$. Now, define $C' = (c_1', c_2', \ldots, c_m')$. We further let $\hat{C} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_q)$ denote the CNT obtained from $C'$ by removing all of the CNOs $c = (\ell, h, w)$ such that $h < \ell$. Denote $\hat{d} = op(\hat{C}, -1, \mathrm{prev}(i))$. Observe that $\hat{d} \leq B$ and that $\hat{C}$ is a $(\mathrm{prev}(i), \hat{d})$-CNT (because $C$ is an $(i, d)$-CNT). Therefore, by the induction hypothesis, $\mathrm{M}[\mathrm{prev}(i), \hat{d}] \leq q$ (recall that $q = |\hat{C}|$). If $\mathrm{prev}(i) = i - 1$, then $Q_i = 0$ and since $C$ is ordered and elongated, by Observation 1 we have that $m - q = \max\{d - \hat{d}, 0\} + \max\{a(i, d) - a(prev(i), \hat{d}), 0\}$. Thus, by the recursive formula, in this case we get that $\mathrm{M}[i, d] \leq m$.

Now, suppose that $\mathrm{prev}(i) < i - 1$. Then, since $C$ is ordered and zero-skipping, and by the definition of $Q_i$, the two following conditions hold.

1. $op(C, -1, i-1) = \max\{Q_i, op(C, -1, \mathrm{prev}(i))\}$.
2. $op(C, 1, i-1) = op(C, 1, \mathrm{prev}(i))$.

Thus, since $C$ is ordered and elongated, by Observation 1 we have that $m - q = \max\{d - \hat{d}, 0\} + \max\{a(i, d) - a(\mathrm{prev}(i), \hat{d}), 0\} + \max\{Q_i - \max\{d, \hat{d}\}, 0\}$. Again, by the recursive formula, this implies that $\mathrm{M}[i, d] \leq m$.

**Second Direction:** Next, we show that $\mathrm{M}[i, d] \geq m$. To this end, it is sufficient to show that there exists an $(i, d)$-CNT $C$ such that $\mathrm{M}[i, d] \geq |C|$. Let $\hat{d}$ be an argument $d'$ at which the value computed by using the recursive formula is minimized. By the inductive hypothesis, there exists a $(\mathrm{prev}(i), \hat{d})$-CNT $\hat{C} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_q)$ such that $\mathrm{M}[\mathrm{prev}(i), \hat{d}] \geq q$. For all $1 \leq j \leq q$, denote $\hat{c}_j = (\ell_j, h_j, w_j)$. Now, if $\mathrm{prev}(i) = i - 1$, define $\tilde{C} = \hat{C}$, else define $\tilde{C}$ as follows. For all $1 \leq j \leq q$, let $\tilde{c}_j = (\ell_j, \tilde{h}, w_j)$, where $\tilde{h} = h_j$ if $h_j < \mathrm{prev}(i)$ and $\tilde{h} = i - 1$ otherwise. Let $\tilde{C} = (\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_q)$. Moreover, as long as there exists $\mathrm{prev}(i) < j < i$ such that $op(\tilde{C}, -1, j) < s_j$, choose the smallest such $j$, and append to the beginning of $\tilde{C}$ the CNO $(j, i-1, -1)$. Let $C'$ be the CNT obtained at the end of this process. Denote $C' = (c_1', c_2', \ldots, c_r')$, and for all $1 \leq j \leq r$, denote $c_j' = (\ell_j', h_j', w_j')$. Now, let $p$ and $q$ be the number of deletions and amplifications in $C'$ whose segments include $i - 1$, respectively. If $p < d$, append to the beginning of $C'$ $d - p$ "dummy" deletions of the form $(i, i-1, -1)$, and if $a(i, d) < q$, append to the end of $C'$ $a(i, d) - q$ "dummy" amplifications of the form $(i, i-1, 1)$. Let $C'' = (c_1'', c_2'', \ldots, c_k'')$ be the resulting CNT, and for all $1 \leq j \leq k$, denote $c_j'' = (\ell_j'', h_j'', w_j'')$. Finally, we define $C$ as follows. Let $D$ $(A)$ be a set of exactly $d$ deletions (resp. amplifications) in $C''$ whose second argument is $i - 1$. We let $C$ be defined as $C''$, except that each CNO $(\ell, h, w) \in D \cup A$ is replaced by the CNO $(\ell, i, w)$. It is straightforward to verify that $C$ is an $(i, d)$-CNT such that $|C| = q + \max\{d - \hat{d}, 0\} + \max\{a(i, d) - a(\mathrm{prev}(i), \hat{d}), 0\} + \max\{Q_i - \max\{d, \hat{d}\}, 0\}$, which concludes the correctness of the second direction.

Now, we turn to consider the correctness and running time of `DpCntpAlg`.

**Theorem 1.** `DpCntpAlg` *solves* CNTP *in time* $O(nB^2)$ *and space* $O(B)$.

*Proof.* The table M contains $O(nB)$ entries, and each entry can be computed in time $O(B)$. Therefore, the time complexity of `DpCntpAlg` is bounded by $O(nB^2)$. Moreover, for the computation of $\mathrm{M}[i, \cdot]$, it is only necessary to keep $O(B)$ entries for position $\mathrm{prev}(i)$, and therefore, the space complexity is bounded by $O(B)$. Since every $(n, d)$-CNT $C$ satisfies $C(S) = T$, and since for every good optimal CNT $C$, there exists $0 \leq d \leq B$ such that $C$ is an $(n, d)$-CNT, we have that Lemma 1 implies that `DpCntpAlg` returns the smallest size of a good optimal CNT (if such a CNT exists). By Propositions 1–4, such a CNT indeed exists, and therefore `DpCntpAlg` solves CNTP. ■

## *3.3. A linear-time algorithm for CNTP*

In this section we show how to modify `DpCntpAlg` to obtain an algorithm, called LinearCntpAlg, that solves CNTP in linear time. The central lemma that leads to this improvement states that each column in the table M can be described by a piecewise linear function of at most three segments.

To present this lemma, we need the following notation. For all $i \in \{1, 2, \ldots, n\}$ such that $t_i > 0$, let $d_i^{min} = \max\{u_i, 0\}$ and $d_i^{max} = \max\{s_i - 1, 0\}$ be the least and largest values of $d$ for which $M[i, d]$ is finite. Now, the function $f_i : \{d_i^{min}, \ldots, d_i^{max}\} \to \mathbb{N} \cup \{0\}$ will satisfy $f_i(d) = \mathrm{M}[i, d]$. Observe that the function $f_i$ is discrete. We stress that in this section, we do not explicitly compute the entries of M—the definition of the functions concerns the values that would have been stored in these entries if they were computed by using `DpCntpAlg`.

**Lemma 2.** *For each $i \in \{1, 2, \ldots, n\}$ such that $t_i > 0$, there exist $base_i, a_i, b_i \in \mathbb{N} \cup \{0\}$ such that for all $d \in \{d_i^{min}, \ldots, d_i^{max}\}$:*

$$f_i(d) = \begin{cases} base_i & \text{if} \quad d_i^{min} \le d \le a_i \\ (base_i - a_i) + d & \text{if} \quad a_i \le d \le b_i \\ (base_i - a_i - b_i) + 2d & \text{if} \quad b_i \le d \le d_i^{max} \end{cases}$$

*Moreover, $base_1, a_1$ and $b_1$ can be computed in constant time, and for each $i \in \{2, 3, \ldots, n\}$ such that $t_i > 0$, given $base_{\mathrm{prev}(i)}, a_{\mathrm{prev}(i)}$ and $b_{\mathrm{prev}(i)}$, $base_i$, $a_i$ and $b_i$ can be computed in constant time.*

An example is given in Figure 7. The proof is based on Lemma 1 and on an exhaustive case analysis, which, for the sake of clarity of presentation, is handled separately in Section 3.4.

Our algorithm, `LinearCntpAlg`, performs the following computation, using `PiecewiseAlg`, an algorithm that computes $base_i$, $a_i$, and $b_i$ in constant time. That algorithm is described in the next subsection.

We are now ready to prove our main result.

**Algorithm 2:** `LinearCntpAlg`

```
Input: $S$, $T$, $Q$, $prev$
Output: $dist(S, T)$
  $base_0 \leftarrow 0$; $a_0 \leftarrow 0$; $b_0 \leftarrow 0$.
  for $i = 1, \ldots, n$, $t_i > 0$ do
    $base_i$; $a_i$, $b_i \leftarrow$ PiecewiseAlg($s_i$, $t_i$, $Q_i$, $base_{prev(i)}$, $a_{prev(i)}$, $b_{prev(i)}$).
  end for
  return $base_n$
```

**Theorem 2.** `LinearCntpAlg` *solves* CNTP *in time $O(n)$ and space $O(1)$.*

*Proof.* According to Lemma 2, $f_i(d) = \mathrm{M}[i, d]$ is a piecewise linear function described by three values: $base_i$, $a_i$ and $b_i$. Lemma 2 shows that `PiecewiseAlg` calculates these values in constant time and space given the previous values. The time and space complexity of `LinearCntpAlg` follow directly. ■

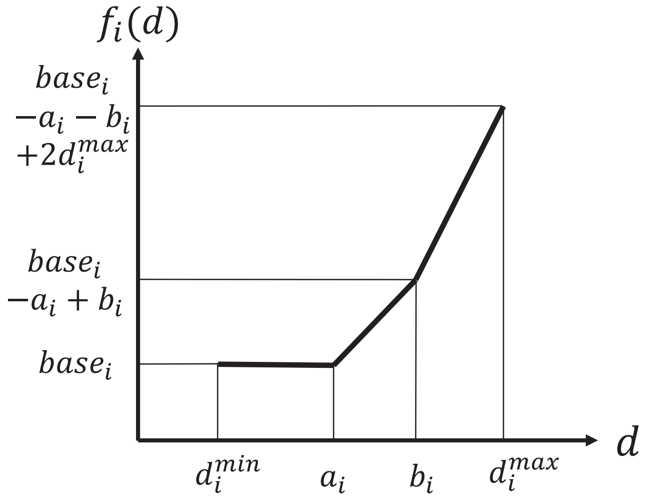

**FIG. 7.** An example of the piecewise linear function $f_i(d)$ described in Lemma 2. The number of segments is three but can be smaller, depending on the values involved.

Now, by the correctness of `DpCntpAlg`, it is sufficient to prove that `LinearCntpAlg` returns the value $\min_{0 \le d \le B} \mathrm{M}[n, d]$. By Observation 2, $\min_{0 \le d \le B} \mathrm{M}[n, d] = \min_{d_n^{min} \le d \le d_n^{max}} \mathrm{M}[n, d]$. By Lemma 2, we further have that $\min_{d_n^{min} \le d \le d_n^{max}} \mathrm{M}[n, d] = base_n$. Thus, by the inductive proof of Lemma 2, we conclude that `LinearCntpAlg` solves CNTP. ■

## 3.4. *Case analysis*

This section is to prove the correctness of Lemma 2. That is, we want to show that $f_i(d)$ is a piecewise linear function described by three parameters, and these parameters can be calculated in constant time. To this end, let $j = \mathrm{prev}(i)$ and $R_i = u_j - u_i$. Accordingly, the term $a(i, d) - a(j, d')$ can be written as $R_i + d - d'$. Moreover, let $d'_{opt}$ be the argument $d'$ that minimizes the recursive formula we use to compute $\mathrm{M}[i, d]$ under certain conditions that will be clear from context.

We prove Lemma 2 by induction on $i$. To simplify the proof, let $a_0 = b_0 = base_0 = 0$ and $f_0(d) = 2d$ for every $0 \le d \le B$. This definition is equivalent to adding the new entries $s_0 = t_0 = B + 1$ (which do not affect the distance from $S$ to $T$), and thus, it can serve as the basis of our induction. Next, suppose that Lemma 2 holds for $j = \mathrm{prev}(i) < i$, we will prove that it holds for $i$.

The proof is based on an exhaustive case analysis that examines the position of $Q_i$ relative to $d_j^{min}$, $a_j$, $b_j$, and $d_j^{max}$, as well as the sign of $R_i$. For example, Case 2(a)ii is defined by the conditions $d_j^{min} \le Q_i \le a_j$, $R_i \ge 0$, and $a_j - R_i \le Q_i$. In each case, we analyze the behavior of $\mathrm{M}[i, d]$ as we increase $d$. More precisely, we examine several intervals that together contain all of the values that can be assigned to $d$. For example, in the abovementioned case, we consider the intervals $d \le a_j - R_j$, $a_j - R_j \le d \le Q_i$, and $Q_i \le d$. For each interval, we let $d'_{opt}$ be an argument $d'$ that minimizes $\mathrm{M}[i, d]$ under the conditions of the examined case. These conditions along with $d'_{opt}$ allow us to remove the minimization and maximization functions from the formula defining $\mathrm{M}[i, d]$, and thus, we obtain $f_i(d)$. In the latter example, if $d \le a_j - R_j$ we can choose $d'_{opt} = a_j$ and get $f_i(d) = \mathrm{M}[i, d] = \mathrm{M}[j, a_j] + \max\{d - a_j, 0\} + \max\{R_i + d - a_j, 0\} + \max\{Q_i - \max\{d, a_j\}, 0\}\} = base_j$. As a corollary of the analysis, we get that indeed $f_i(d)$ is piecewise linear, and that $a_i$, $b_i$, and $base_i$ can be calculated in constant time given $a_j$, $b_j$, $base_j$, $R_i$, and $Q_i$.

The full case analysis is given in the Appendix. The analysis shows that in all cases, $f_i(d)$ is indeed a piecewise linear function with at most three linear segments defined by some $a_i$, $b_i$, and $base_i$. After applying straightforward operations that reorganize the analysis (to present the results in a compact manner), we obtain the algorithm `PiecewiseAlg`, whose pseudocode is given below. This algorithm performs the iterative step of `LinearCntpAlg`, that is, it calculates $a_i$, $b_i$, $base_i$ given $a_j$, $b_j$, $base_j$, and $Q_i$ in constant time and space.

`PiecewiseAlg` first calculates $R_i$, $d_i^{min}$ and $d_i^{max}$ based on $s_i$ and $t_i$. Next, according to the sign of $R_i$ and the relative position of $Q_i$ in comparison to the previous $a_j$ and $b_j$, the algorithm calculates the structure of $f_i(d)$ defined by $a_i$ and $b_i$. Finally, since $f_i(d)$ is defined only for the range $d_i^{min} \le d \le d_i^{max}$, we calculate $base_i = f_i(d_i^{min})$. Similarly, we limit the values of $a_i$ and $b_i$ to that range.

**Algorithm 3:** `PiecewiseAlg`

```
Input: $s_i, t_i, Q_i, a_j, b_j, base_j$
Output: $a_i, b_i, base_i$
  $R_i \leftarrow u_j - u_i$
  $d_i^{min} \leftarrow \max\{u_i, 0\}$
  $d_i^{max} \leftarrow \max\{s_i - 1, 0\}$
  $a_i \leftarrow \min\{\max\{a_j, Q_i\}, b_j - \min\{R_i, 0\}\} - \max\{R_i, 0\}$
  $b_i \leftarrow \max\{Q_i, b_j - \min\{R_i, 0\}\}$
  $base_i \leftarrow base_j + \max\{Q_i - a_j, 0\} + \begin{cases} 0 & \text{if } d_i^{min} \le a_i \\ d_i^{min} - a_i & \text{if } a_i < d_i^{min} \le b_i \\ 2d_i^{min} - a_i - b_i & \text{if } b_i < d_i^{min} \le d_i^{max} \end{cases}$
  $a_i \leftarrow \max\{d_i^{min}, \min\{a_i, d_i^{max}\}\}$; $b_i \leftarrow \max\{a_i, \min\{b_i, d_i^{max}\}\}$
  return $base_i, a_i, b_i$
```

## 4. CONCLUSION

In this article, we introduced the study of distances between CNPs from a theoretical point of view. We focused on one fundamental problem, CNTP, and showed that it is solvable in linear time and constant space. To this end, we proved several properties of CNTP that may be useful in solving other problems involving CNPs. Our algorithm can be modified to return a transformation that realizes $dist(S, T)$ in linear time and linear space by backtracking the dynamic programming vector. We have implemented the algorithm as well as a linear programming formulation of CNTP, and the implementations are available on request.

Many computational and combinatorial aspects in the analysis of distances between CNPs require further research. Indeed, this article can be viewed as a first step toward understanding them. In our follow-up article by El-Kebir et al. (2016), we investigated a generalization of CNTP where the input is a set of profiles, and one seeks to construct a tree with the profile labels at the leaves and additional profile labeling of internal nodes that minimizes the transformation distances along the edges. We showed this problem is NP-hard and gave an Integer Linear Programming (ILP) formulation to solve it. Additional directions for further research involve the introduction of edit operations other than basic segmental deletions and amplifications, dealing with phasing of the profiles, as well as the handling of noise.

## ACKNOWLEDGMENTS

We thank the referees for many helpful comments. This study was supported by the Israeli Science Foundation (grant 317/13), the Israel Cancer Association, and the Dotan Hemato-Oncology Research Center at Tel Aviv University. R.Z. was supported by fellowships from the Edmond J. Safra Center for Bioinformatics at Tel Aviv University and from the Israeli Center of Research Excellence (I-CORE) Gene Regulation in Complex Human Disease (Center No 41/11). M.Z. was supported by a fellowship from the I-CORE in Algorithms and the Simons Institute for the Theory of Computing in Berkeley and by the Postdoctoral Fellowship for Women of Israel's Council for Higher Education.

## AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

## REFERENCES

Abo, R.P., Ducar, M., Garcia, E.P., et al. 2015. BreaKmer: Detection of structural variation in targeted massively parallel sequencing data using kmers. *Nucleic Acids Res.* 18; 43(3):e19.

Chowdhury, S.A., Gertz, E.M., Wangsa, D., et al. 2015. Inferring models of multiscale copy number evolution for single-tumor phylogenetics. *Bioinformatics* 31, i258–i267.

Chowdhury, S.A., Shackney, S.E., Heselmeyer-Haddad, K., et al. 2013. Phylogenetic analysis of multiprobe fluorescence in situ hybridization data from tumor cell populations. *Bioinformatics* 29, i189–i198.

Chowdhury, S.A., Shackney, S.E., Heselmeyer-Haddad, K., et al. 2014. Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLoS Comput. Biol.* 10, e1003740.

El-Kebir, M., Raphael, B.J., Shamir, R., et al. 2016. *Copy-Number Evolution Problems: Complexity and Algorithms*, pages 137–149. Springer International Publishing, Cham.

Fertin, G., Labarre, A., Rusu, I., et al. 2009. *Combinatorics of Genome Rearrangements*. MIT Press, Cambridge, MA.

Letouzé, E., Allory, Y., Bollet, M.A., et al. 2010. Analysis of the copy number profiles of several tumor samples from the same patient reveals the successive steps in tumorigenesis. *Genome Biol.* 11, R76.

McPherson, A., Wu, C., Wyatt, A.W., et al. 2012. nFuse: Discovery of complex genomic rearrangements in cancer using high-throughput sequencing. *Genome Res.* 22, 2250–2261.

Mohri, M. 2003. Edit-distance of weighted automata: General definitions and algorithms. *Int. J. Found. Comput. Sci.* 14, 957–982.

Mohri, M. 2004. Weighted finite-state transducer algorithms. An overview. *In Formal Languages and Applications*. pp. 551–563. Springer, Berlin-Heidelberg.

Oesper, L., Ritz, A., Aerni, S.J., et al. 2012. Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinformatics* 13 Suppl 6, S10.

Pinkel, D., Straume, T., and Gray, J.W. 1986. Cytogenetic analysis using quantitative, high-sensitivity, fluorescence hybridization. *Proc. Natl Acad. Sci. U. S. A.* 83, 2934–2938.

Savard, O.T., Gagnon, Y., Bertrand, D., et al. 2011. Genome halving and double distance with losses. *J. Comput. Biol.* 18, 1185–1199.

Schwarz, R.F., Trinh, A., Sipos, B., et al. 2014. Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Comput. Biol.* 10, e1003535.

Shamir, R., Zehavi, M., and Zeira, R. 2016. A linear-time algorithm for the copy number transformation problem. *In* Grossi, R., and Lewenstein, M., eds, *27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)*, volume 54 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Shao, M., and Lin, Y. 2012. Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. *BMC Bioinformatics* 13, S13.

Tannier, E., Zheng, C., and Sankoff, D. 2009. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics* 10, 120.

The Cancer Genome Atlas Research Network. 2011. Integrated genomic analyses of ovarian carcinoma. *Nature* 474, 609–615.

Urban, A.E., Korbel, J.O., Selzer, R., et al. 2006. High-resolution mapping of DNA copy alterations in human chromosome 22 using high-density tiling oligonucleotide arrays. *Proc. Natl Acad. Sci. U. S. A.* 103, 4534–4539.

Address correspondence to:
*Ron Zeira*
*Blavatnik School of Computer Science*
*Tel-Aviv University*
*Tel-Aviv 69978*
*Israel*

*E-mail:* ronzeira@post.tau.ac.il

## 5. APPENDIX

### 5.1. Detailed case analysis

In this appendix, we present the details of the case analysis outlined in Section 3.4. We analyze the behavior of M$[i, d]$ as we increase $d$. We assume, by induction, that $f_j(d)$ is a piecewise linear function with parameters $a_j$, $b_j$ and $base_j$ for $j = \text{prev}(i)$. Then, we examine several intervals that together contain all of the values that can be assigned to $d$. For each interval, we let $d'_{opt}$ be an argument $d'$ that minimizes M$[i, d]$ under the conditions of the examined case. Finally, we obtain the behavior of $f_i(d)$ in each interval, which is the behavior of the form we desire (i.e., $f_i(d)$ is a piecewise linear function defined by three segments). Denote $\max\{Q_i - \max\{d, d'\}, 0\}$ as $\text{arg}_3$.

1. $Q_i \leq d_j^{min}$(then, $\text{arg}_3 = 0$):
   (a) $R_i \geq 0$:
      i. $d \leq a_j - R_i$:
         $d'_{opt} = a_j : f_i(d) = base_j$.
      ii. $a_j - R_i \leq d \leq b_j$:
         $d'_{opt} = d : f_i(d) = base_j + R_i - a_j + d$.
      iii. $b_j \leq d \leq d_j^{max}$:
         $d'_{opt} = d : f_i(d) = base_j + R_i - a_j - b_j + 2d$.
      iv. $d_j^{max} \leq d$:
         $d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d$.

   (b) $R_i \leq 0$:
      i. $d \leq a_j + R_i$:
         $d'_{opt} = a_j : f_i(d) = base_j$.
      ii. $a_j + R_i \leq d \leq a_j + R_i$:
         $d'_{opt} = d : f_i(d) = base_j$.
      iii. $a_j \leq d \leq b_j$:
         $d'_{opt} = d : f_i(d) = base_j - a_j + d$.

iv. $b_j \leq d \leq b_j - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j - a_j + d$.
v. $b_j - R_i \leq d \leq d_j^{max} - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + R_i - a_j - b_j + 2d$.
vi. $d_j^{max} - R_i \leq d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d$.

2. $d_j^{min} \leq Q_i \leq a_j$:
(a) $R_i \geq 0$:
i. $Q_i \leq a_j - R_i$ : $\arg_3 = 0$ and the analysis is the same as in Case 1a.
ii. $a_j - R_i \leq Q_i$:
A. $d \leq a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j$.
B. $a_j - R_i \leq d \leq Q_i$:
$d'_{opt} = a_j : f_i(d) = base_j + R_i - a_j + d$.
C. $Q_i \leq d$ : $\arg_3 = 0$ and the rest of the analysis is the same as in Case 1a.

(b) $R_i \leq 0$:
i. $Q_i \leq a_j + R_i$ : $\arg_3 = 0$ and the analysis is the same as in Case 1b.
ii. $a_j + R_i \leq Q_i$:
A. $d \leq a_j + R_i$:
$d'_{opt} = a_j : f_i(d) = base_j$.
B. $a_j + R_i \leq d \leq Q_i$:
$d'_{opt} = a_j : f_i(d) = base_j$.
C. $Q_i \leq d$ : $\arg_3 = 0$ and the rest of the analysis is the same as in Case 1b.

3. $a_j \leq Q_i \leq b_j$:
(a) $R_i \geq 0$:
i. $d \leq a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$.
ii. $a_j - R_i \leq d \leq Q_i - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j$.
iii. $Q_i - R_i \leq d \leq Q_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + R_i - a_j + d$.
iv. $Q_i \leq d \leq b_j$:
$d'_{opt} = d : f_i(d) = base_j + R_i - a_j + d$.
v. $b_j \leq d \leq d_j^{max}$:
$d'_{opt} = d : f_i(d) = base_j + R_i - a_j - b_j + 2d$.
vi. $d_j^{max} \leq d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d$.

(b) $R_i \leq 0$:
i. $d \leq a_j$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$.
ii. $Q_i \leq a_j - R_i$:
A. $a_j \leq d \leq Q_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i - a_j$.
B. $Q_i \leq d \leq a_j - R_i$:
$d'_{opt} = d : f_i(d) = base_j - a_j + d$.
iii. $a_j - R_i \leq Q_i$:
A. $a_j \leq d \leq a_j - R_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i - a_j$.
B. $a_j - R_i \leq d \leq Q_i$:
$d'_{opt} = d : f_i(d) = base_j + Q_i - a_j$.
iv. $Q_i \leq d \leq b_j$:
$d'_{opt} = d : f_i(d) = base_j - a_j + d$.

v. $b_j \le d \le b_j - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j - a_j + d$.

vi. $b_j - R_i \le d \le d_j^{max} - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + R_i - a_j - b_j + 2d$.

vii. $d_j^{max} - R_i \le d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + R_i - a_j - b_j + 2d$.

4. $b_j \le Q_i \le d_j^{max}$:
   (a) $R_i \ge 0$:
      i. $d \le a_j - R_i$:
      $d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$.
      ii. $Q_i - R_i \le a_j$:
         A. $a_j - R_i \le d \le b_j - R_i$:
         $d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j$.
         B. $b_j - R_i \le d \le Q_i - R_i$:
         $d'_{opt} = b_j : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d$.
         C. $Q_i - R_i \le d \le Q_i$:
         $d'_{opt} = d : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d$.
      iii. $a_j \le Q_i - R_i \le b_j$:
         A. $a_j - R_i \le d \le b_j - R_i$:
         $d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j$.
         B. $b_j - R_i \le d \le Q_i - R_i$:
         $d'_{opt} = d + R_i : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d$.
         C. $Q_i - R_i \le d \le Q_i$:
         $d'_{opt} = d : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d$.
      iv. $b_j \le Q_i - R_i$:
         A. $a_j - R_i \le d \le b_j - R_i$:
         $d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j$.
         B. $b_j - R_i \le d \le Q_i - R_i$:
         $d'_{opt} = d + R_i : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d$.
         C. $Q_i - R_i \le d \le Q_i$:
         $d'_{opt} = d : f_i(d) = base_j + Q_i + R_i - a_j - b_j + d$.
      v. $Q_i \le d$:
      $d'_{opt} = Q_i : f_i(d) = base_j + R_i - a_j - b_j + 2d$.
   (b) $R_i \le 0$:
      i. $d \le a_j$:
      $d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$.
      ii. $Q_i \le a_j - R_i$:
         A. $a_j \le d \le Q_i$:
         $d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$.
         B. $Q_i \le d \le a_j - R_i$:
         $d'_{opt} = a_j : f_i(d) = base_j - a_j + d$.
         C. $a_j - R_i \le d \le b_j - R_i$:
         $d'_{opt} = a_j : f_i(d) = base_j - a_j + d$.
         D. $b_j - R_i \le d \le Q_i - R_i$:
         $d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d$.
      iii. $a_j - R_i \le Q_i \le b_j - R_i$:
         A. $a_j \le d \le a_j - R_i$:
         $d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$.
         B. $a_j - R_i \le d \le Q_i$:
         $d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j$.
         C. $Q_i \le d \le b_j - R_i$:
         $d'_{opt} = d + R_i : f_i(d) = base_j - a_j + d$.
         D. $b_j - R_i \le d \le Q_i - R_i$: $d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d$.

iv. $b_j - R_i \leq Q_i$:
A. $a_j \leq d \leq a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$
B. $a_j - R_i \leq d \leq b_j - R_i$:
$d'_{opt} = p + R_i : f_i(d) = base_j + Q_i - a_j$.
C. $b_j - R_i \leq d \leq Q_i$:
$d'_{opt} = b_j : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
D. $Q_i \leq d \leq Q_i - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d$.
v. $Q_i - R_i \leq d$:
$d'_{opt} = b_j : f_i(d) = base_j + M_i - a_j - b_j + 2d$.

5. $d_j^{max} \leq Q_i$:
(a) $R_i \geq 0$:
i. $Q_i - R_i \leq d_j^{max}$: The analysis is the same as in Case 4a for $d \leq d_j^{max}$.
A. $d_j^{max} \leq d \leq Q_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
ii. $Q_i - R_i \leq d_j^{max}$:
A. $d \leq b_j - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + Q_i - a_j$.
B. $b_j - R_i \leq d \leq d_j^{max} - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
C. $d_j^{max} - R_i \leq d \leq d_j^{max}$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
D. $d_j^{max} \leq d \leq Q_i - R_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
E. $Q_i - R_i \leq d \leq Q_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
iii. $Q_i \leq d$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i - a_j - b_j + 2d$.

(b) $R_i \leq 0$:
i. The analysis of the cases obtained by adding the constraints defining Cases 4(b)ii, 4(b)iii, and 4(b)iv is similar.
ii. $d_j^{max} \leq Q_i \leq d_j^{max} - R_i$:
A. $d \leq a_j - R_i$:
$d'_{opt} = a_j : f_i(d) = base_j + Q_i - a_j$.
B. $a_j - R_i \leq d \leq b_j - R_i$:
$d'_{opt} = b_j : f_i(d) = base_j + Q_i - a_j$.
C. $b_j - R_i \leq d \leq d_j^{max}$:
$d'_{opt} = d : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
D. $d_j^{max} \leq d \leq Q_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
E. $Q_i \leq d \leq d_j^{max} - R_i$:
$d'_{opt} = d_j^{max} + R_i : f_i(d) = base_j + M_i - a_j - b_j + 2d$.
F. $d_j^{max} - R_i \leq d$:
$d'_{opt} = d_j^{max} + R_i : f_i(d) = base_j + M_i - a_j - b_j + 2d$.
iii. $d_j^{max} - R_i \leq Q_i$:
A. For $d \leq d_j^{max}$, the analysis remains the same as in Case 5(b)ii.
B. $d_j^{max} \leq d \leq d_j^{max} - R_i$:
$d'_{opt} = d + R_i : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
C. $d_j^{max} - R_i \leq d \leq Q_i$:
$d'_{opt} = d_j^{max} : f_i(d) = base_j + M_i + Q_i - a_j - b_j + d$.
D. $Q_i \leq d$:
$d'_{opt} = d_j^{max} + R_i : f_i(d) = base_j + M_i - a_j - b_j + 2d$.

# Chapter 4

# Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions

*Bioinformatics*, 2018, 1–8
doi: 10.1093/bioinformatics/bty381
Advance Access Publication Date: 3 May 2018
Original Paper

Genome analysis

# Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions

**Ron Zeira* and Ron Shamir***

Blavatnik School of Computer Science, Tel Aviv university, Tel Aviv 6997801, Israel

*To whom correspondence should be addressed.

Associate Editor: Oliver Stegle

Received and revised on March 15, 2018; editorial decision on April 27, 2018; accepted on May 2, 2018

## Abstract

**Motivation:** Problems of genome rearrangement are central in both evolution and cancer research. Most genome rearrangement models assume that the genome contains a single copy of each gene and the only changes in the genome are structural, i.e. reordering of segments. In contrast, tumor genomes also undergo numerical changes such as deletions and duplications, and thus the number of copies of genes varies. Dealing with unequal gene content is a very challenging task, addressed by few algorithms to date. More realistic models are needed to help trace genome evolution during tumorigenesis.
**Results:** Here, we present a model for the evolution of genomes with multiple gene copies using the operation types double-cut-and-joins, duplications and deletions. The events supported by the model are *reversals*, *translocations*, *tandem duplications*, *segmental deletions* and *chromosomal amplifications* and *deletions*, covering most types of structural and numerical changes observed in tumor samples. Our goal is to find a series of operations of minimum length that transform one karyotype into the other. We show that the problem is NP-hard and give an integer linear programming formulation that solves the problem exactly under some mild assumptions. We test our method on simulated genomes and on ovarian cancer genomes. Our study advances the state of the art in two ways: It allows a broader set of operations than extant models, thus being more realistic and it is the first study attempting to re-construct the full sequence of structural and numerical events during cancer evolution.
**Availability and implementation:** Code and data are available in https://github.com/Shamir-Lab/Sorting-Cancer-Karyotypes.
**Contact:** ronzeira@post.tau.ac.il or rshamir@tau.ac.il
**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 Introduction

During cancer, the tumor genome rapidly accumulates somatic mutations. While some mutations are small, affecting one or a few bases, others are large-scale events. Here, we focus on the latter. They include inversions, chromosomal translocations, tandem duplications, segmental deletions and whole chromosome amplifications or losses (Vogelstein *et al.*, 2013). Some cancer types are predominately characterized by these types of mutations (Ciriello *et al.*, 2013). Understanding these changes can assist in predicting disease progression and the outcome of medical interventions (Fielding, 2010). For instance, early translocations and tandem duplications in ovarian cancer were shown to contribute to drug sensitivity and clonal expansion (Ng *et al.*, 2012).

### 1.1 Aberration types and cancer genome data

The *copy number* (*CN*) of a genomic segment is the number of copies of the segment a genome contains. In a healthy diploid genome, each segment has $CN = 2$. In a *segmental deletion*, a segment of the DNA is deleted resulting in a genome with one less copy of the segment. In *chromosomal deletion*, an entire chromosome is deleted.

In a *tandem duplication*, a segment of a chromosome is duplicated and inserted right after the original one. A *chromosomal duplication* (or *amplification*) creates an additional copy of an entire chromosome. Overall, deletions and duplications can change both the structure and the CN of the genome.

Other aberrations change only the structure of the genome but not its CNs. In an *inversion* (or *reversal*), a segment of a chromosome is reversed relative to its original orientation. In a *translocation*, two chromosomes exchange ends segments.

Given the germline genome $G$ and the tumor genome $T$, a *breakpoint* is a position between two bases that are consecutive in $G$ but not in $T$. Inversions and translocations introduce two breakpoints, segmental deletions and tandem duplications introduce one breakpoint and chromosomal duplications/deletions introduce no breakpoints.

The primary source of data for cancer genome analysis today is deep sequencing. It allows inference of CN changes based on read depth (Ding *et al.*, 2014), and facilitates inferring breakpoints in the genome, detecting structural variants and identifying rearrangements (Korbel *et al.*, 2007). If the mapped locations of the two ends of a paired-end read do not match the read length, the read is called *discordant* and suggests a breakpoint in the genome. The location and orientation of such discordant reads can help detect the type of event (Abo *et al.*, 2015). Accurate re-construction of the numerical and structural variations from deep sequencing data remain a challenge, and a myriad of computational methods have been devised for this task (Ding *et al.*, 2014).

## 1.2 Genome rearrangement models

Over the past two decades, many genome rearrangement models were studied. The classical model seeks a shortest sequence of inversions and translocations that transform one genome into another (Hannenhalli and Pevzner, 1996, 1999). Such a sequence is called a *sorting scenario*. Later, a simpler model based on *double-cut-and-join* (*DCJ*) was proposed. In a DCJ, the genome is cut in two locations and the four loose ends are re-connected as two pairs. This model can represent both inversions and translocations (Bergeron *et al.*, 2006; Yancopoulos *et al.*, 2005). Feijão and Meidanis (2011) provided a simpler model called *single-cut-or-join* (*SCoJ*), in which every operation either cuts the genome or joins two loose ends. These DCJ and SCoJ models assumed a single copy of each genomic segment and no operation that alters CN. Extant models with multiple segment copies often result in NP-hard problems (Shao and Lin, 2012; Tannier *et al.*, 2009). Some rearrangement models assume that a breakpoint cannot be used twice in a sorting scenario (Pevzner and Tesler, 2003).

Several models have addressed multiple copies along with other operation types. Some allow insertions or deletions of genomic segments along with DCJs, but only for non-duplicated segments (da Silva *et al.*, 2012). For sorting multiple copy genomes using DCJs only, both an exact integer linear program (ILP) and an approximation have been given (Shao *et al.*, 2015; Shao and Lin, 2012). Bader (2010) provided a heuristic for sorting by DCJs, duplications and deletions. Shao and Moret (2015) devised an ILP for sorting genomes with multiple copies via DCJs and certain type of segmental duplications. Zeira and Shamir (2017) gave a linear algorithm for sorting with SCoJs and chromosomal duplications on genomes with at most two copies. Ozery-Flato and Shamir (2009) studied a model with certain duplications, deletions and SCoJs and provided a three-approximation algorithm that performed well on cancer genomes.

Several models attempted to introduce CN-modifying operations. Chowdhury *et al.* (2014) defined an edit distance between CN profiles obtained from FISH, where the edit operations are amplification or deletion of single genes, single chromosomes, or the whole genome. However, these methods are tailored to FISH data with a limited number of genes. Schwarz *et al.* (2014) introduced a model that allows amplifications and deletions of contiguous segments. A linear time algorithm for this edit distance was later given (Zeira *et al.*, 2017). However, all these models consider only CN modifications but not structural rearrangements.

## 1.3 Graph models for tumor rearrangements

Graph theory contributed remarkably to the area of genomic rearrangements. Breakpoint graphs are widely used for representation and analysis of rearranged genomes in evolution (Bafna and Pvezner, 1996; Hannenhalli and Pevzner, 1995b) and in cancer genomes (Raphael *et al.*, 2003). Greenman *et al.* (2012) created models that expanded the breakpoint graph and they used them in order to infer some order over tumor mutations.

Oesper *et al.* (2012) further expanded the breakpoint graph with a structure called the *interval adjacency graph*, which represents breakpoints, discordant reads and CN information. Their method, called *PREGO*, uses the number of reads supporting each edge to resolve the CN of genomic segments and identify discordant adjacencies in the tumor genome. Decomposition of this graph into a set of paths corresponds to a set of chromosomes. PREGO was shown to efficiently identify complex rearrangement in ovarian cancer sequencing data. Eitan and Shamir (2017) expanded this model and tested it in extensive simulations and on real cancer data.

## 1.4 Our contribution

We propose here a model for the structural and numerical changes that a genome with multiple segmental copies undergoes. The allowed operations are DCJs, tandem duplications, segmental deletions and whole chromosome duplications and deletions. This model encompasses many of the common aberrations in cancer, and does not preclude breakpoint reuse. However, we restrict both duplications and deletions to simple paths that include at most one copy of each segment. Similarly to Oesper *et al.* (2012), genomes are represented by the CN of each segment and the adjacencies between them. Our goal is to find a shortest series of operations that transform one genome into the other, e.g. a normal genome to an observed tumor genome. Unlike most models, we focus here on finding the actual sequence of events. We show that the problem is NP-hard, give an ILP formulation for solving this problem and apply it on simulated and ovarian tumor data. The algorithm is able to resolve the sequence of events for tumors of average complexity.

The study advances the state of the art in genome rearrangement analysis in cancer in two ways: It allows a broader set of operations than extant models, thus being more realistic and it is also the first model attempting to re-construct the full sequence of structural and numerical events during cancer evolution.

## 2 Materials and methods

In this section, we present our model and formulate an ILP to solve it.

## 2.1 Notation

A genome contains a set $\mathcal{G} = \mathcal{G}^* \cup \mathcal{T}_{\mathcal{G}}$ of entities. $\mathcal{G}^*$ is a set of $n$ *genes*, denoted $1, \ldots, n$. Each gene $g \in \mathcal{G}$ has two *extremities*, a *head*

$g^h$ and a *tail* $g^t$. W.l.o.g., denote $g^t = 2g$ and $g^h = 2g + 1$ for every $g \in \mathcal{G}$. $\mathcal{T}_{\mathcal{G}}$ is a set of special genes called *telomeres*. Each telomere has only one extremity. Telomeres come in pairs distinguished as the *left telomere* and the *right telomere*. A left telomere has only a head and a right telomere has only a tail. The left and right telomeres correspond to the start and end of chromosomes in the real genome. The genes in $\mathcal{G}^*$ are also called *internal* genes.

Denote by $\mathcal{T}$ the set of extremities corresponding to telomeres, and by $\mathcal{E}^* = \{g^t, g^h | g \in \mathcal{G}^*\}$ the set of extremities of internal genes. The set of all extremities is denoted by $\mathcal{E} = \mathcal{E}^* \cup \mathcal{T}$. Throughout, for an extremity $e$ we shall denote by $g(e)$ the gene it belongs to.

A *karyotype* is represented by a pair $K = (\text{cn}, \text{adj})$. $\text{cn} : \mathcal{G} \to \mathbb{N}$ is a gene *CN profile* and $\text{adj} : \mathcal{E} \times \mathcal{E} \to \mathbb{N}$ is an *adjacency CN matrix*, such that $\forall e \in \mathcal{E}, \text{cn}(g(e)) = \sum_{e' \in \mathcal{E}} \text{adj}(e, e')$. Notice that adj is symmetric, and different copies of a gene or adjacency are indistinguishable.

The *karyotype graph* of $K$ is a weighted undirected graph $G = (\mathcal{E}, E, W)$ akin to the interval adjacency graph (Oesper *et al.*, 2012; Fig. 1). The edge set $E = E_I \cup E_A$ consists of *interval edges* $E_I$ and *adjacency edges* $E_A$. Interval edges $E_i = \{(g^t, g^h) | g \in \mathcal{G}^*, \text{cn}(g) > 0\}$ correspond to genes and adjacency edges $E_i = \{(u, v) | u, v \in \mathcal{E}, \text{adj}(u, v) > 0\}$ correspond to adjacencies in the karyotype. The weight $W : E \to \mathbb{N}$ is defined as the CN of the edge, i.e.

$$W(u, v) = \begin{cases} \text{cn}(g) & \text{if } (u, v) = (g^t, g^h) \in E_I \\ \text{adj}(u, v) & \text{if } (u, v) \in E_A \end{cases}$$

*Removing a copy* of an existing edge $(u, v)$ from $G$ results in a new graph in which the CN of $(u, v)$ is lower by 1 and the edge is deleted from the graph if its new CN is zero. Similarly, *adding a copy* of an edge $(u, v)$ to $G$ results in a new graph $G' = (\mathcal{E}, E', W')$ in which the CN of the edge $(u, v)$ increases by 1 if it exists in $G$, or adding the new edge $(u, v)$ with CN = 1.

An *alternating path* is a simple path in $G$ in which odd edges are interval edges and even edges are adjacency edges, or vise-versa.

A *chromosome* in a karyotype is an alternating path starting and ending with telomeres. Note that a karyotype may be decomposable into chromosomes in several ways.

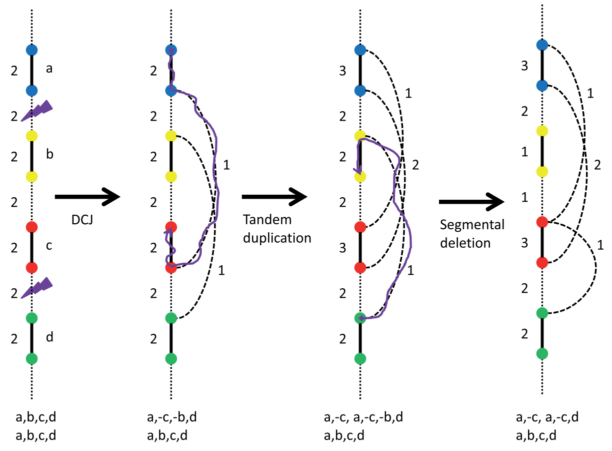

**Fig. 1.** The effect of model operations on the karyotype graph. Solid edges: interval edges; dotted edges: adjacencies; numbers on edges: CN; lightning signs: breakpoints. The scribbled lines show the path affected by the operation. The sequences of genes corresponding to possible chromosomes are shown below each graph, with each chromosome in a separate line. The genes *a,b,c,d* correspond to the segments in the leftmost karyotype graph from top to bottom.

## 2.2 Model

We now turn to define the possible operations that alter a karyotype in our model (compare Fig. 1).

**DCJ:** A *DCJ* operation selects two adjacency edges $(a, b), (c, d)$, removes a copy of each from the graph and adds copies of new edges by joining two loose ends, either $(a, c), (b, d)$ or $(a, d), (b, c)$. In order to support splitting of a chromosome and introducing additional telomere copies, we also allow a special case of DCJ called *Single-cut-and-join* (*SCJ*). SCJ cuts an existing adjacency $(a, b)$ and connects each loose end to a new telomere copy $t_1, t_2 \in \mathcal{T}$. The result of this SCJ is $(a, t_1), (b, t_2)$, i.e. splitting the adjacency into two separate chromosomes capped with new copies of telomeres $t_1$, $t_2$. Note that SCJs create new telomere copies that may or may not be part of the final karyotype. The identity of each $t_i$ can be arbitrary chosen.

**Tandem duplication:** Let $v_1, \ldots, v_{2m}$ be an alternating path without telomeres starting with an interval edge. A *tandem duplication* adds edge copies for each edge in the path and adds another adjacency edge copy $(v_1, v_{2m})$. We call $v_1, v_{2m}$ the *anchors* of the duplication. In terms of the sequence, this operation corresponds to $\ldots g_0 \underline{g_1 g_2 \cdots g_{m-1} g_m} g_{m+1} \cdots \to \cdots g_0 \underline{g_1 g_2 \cdots g_{m-1} g_m} \quad \underline{g_1 g_2 \cdots g_{m-1} g_m} g_{m+1} \ldots$ where $g_i$ is the gene corresponding to node $v_{2i-1}$.

**Chromosome duplication:** Let $t_0, v_1, \ldots, v_{2m}, t_{2m+1}$ be an alternating path such that $t_0, t_{2m+1}$ are telomeric extremities and $(v_1, v_2)$ is an interval edge. A *chromosome duplication* adds edge copies for each edge along the path and increases the CN of the two telomeres by one.

**Segmental deletion:** Let $v_1, \ldots, v_{2m}$ be an alternating path without telomeres staring with an adjacency edge. A segmental deletion removes edge copies for each edge along the path and adds an adjacency edge copy $(v_1, v_{2m})$. We call $v_1, v_{2m}$ the *anchors* of the deletion. In terms of the sequence, this operation corresponds to $\ldots g_0 \underline{g_1 g_2 \cdots g_{m-1} g_m} g_{m+1} \cdots \to \cdots g_0 \underline{g_1 g_m} g_{m+1} \cdots$.

**Chromosome deletion:** This is a special case of segmental deletion where $v_1$ and $v_{2m}$ are telomeric nodes. We do not add the edge $(v_1, v_{2m})$ and thus it corresponds to deleting an entire chromosome with its telomeres.

**The Karyotype Sorting Problem:** The input is $S$, $T$, $d$, where $S = (s\_\text{cn}, s\_\text{adj})$ is a source karyotype, $T = (t\_\text{cn}, t\_\text{adj})$ is a target karyotype and $d$ is an integer. Our goal is to find a shortest series of $\leq d$ operations transforming $S$ into $T$, or declare that no such sequence exists. An example of a series of operations of length 11 is given in Figure 3.

Notice that the sorting problem is not symmetric. Moreover, there may be a sorting scenario from $S$ to $T$, but not from $T$ to $S$ if, for example, $T$ has lost all copies of some segment in $S$. New material can be gained in the model by duplications (tandem and chromosomal). Telomeres can also be gained by SCJs.

THEOREM 1. The karyotype sorting problem is NP-hard.

PROOF. Let $G = (V, E)$ be a directed graph with $n$ nodes in which all in- and out-degrees are 2. Deciding if a such a graph contains a Hamiltonian cycle is NP-hard (Plesńlk, 1979). Let $(y, x) \in E$ be some edge. Deciding if there is a Hamiltonian path from $x$ to $y$ in $G$ is still NP-hard. We assume w.l.o.g. that $G$ is strongly connected, since otherwise it would not contain a Hamiltonian path from $x$ to $y$. Notice that in that case $G$ is also Eulerian.

We construct a source karyotype $S$ as follows: for each node in $v \in V$ we create a gene $g_v$ with CN = 2 and for each $(u, v) \in E \backslash \{(y, x)\}$ we add one copy of the adjacency $\left(g_u^h, g_v^t\right)$. In addition, we add two genes $w$, $z$ with CN = 1 and connect them with adjacencies $\left(w^h, g_x^t\right)$, $\left(g_y^h, z^t\right)$ of CN = 1. To make it a valid karyotype, we add a left and right telomeres $t_1$, $t_2$ and connect them to $w^t$ and $z^h$, respectively. In other words, $S$ is a karyotype with a single chromosome $t_1, w, x, P_e, y, z, t_2$, where $P_e$ corresponds to some Eulerian path from $x$ to $y$ in $G$. Our target karyotype $T$ would be composed of $n$ single gene chromosomes of the form $t_1, g_v, t_2$ and an additional chromosome $t_1, w, z, t_2$. Namely, each gene will have CN$=1$, for telomeres CN$=n+1$, and the adjacencies would be of the form $(t_1, g_v^t)$ and $(g_v^t, t_2)$ for $v \in V$ plus $(t_1, w^t), (w^h, z^t), (z^h, t_2)$. Notice that all chromosome paths start and end with the same telomeres $t_1$, $t_2$. We will show that there is a sorting scenario of $S$ to $T$ of length $n+1$ if and only if $G$ admits a Hamiltonian path from $x$ to $y$.

Suppose $G$ contains a Hamiltonian path $P = x, v_2, \ldots, v_{n-1}, y$. To construct a sorting scenario, first perform $n$ SCJs for each adjacency of the form $(g_u^h, g_v^t)$ that is not part of $P$, and connect them as $(t_1, g_v^t)$ and $(g_u^t, t_2)$. Now, perform a segmental deletion of the path $g_x, g_{v_2}, \ldots, g_{v_{n-1}}, g_y$ connecting $w^h$ and $z^t$. The total length of the sorting scenario is $n+1$.

For the other direction, suppose there is a sorting scenario with $n+1$ operations from $S$ to $T$. Since each gene in $T$ has CN$=1$, the scenario must contain at least one deletion. Notice that $T$ has $n$ additional copies of the telomeres and the only way to increase the CN of telomeres in the model is either by chromosome duplication or by SCJ. However, each chromosome duplication would require additional deletions to reduce the gene CN to 1 in $T$. We conclude that the sorting scenario must contain $n$ SCJs and one deletion covering all genes $g_v$. Since $w$ and $z$ are adjacent in $T$, the segmental deletion must be anchored at $w^h$ and $z^t$. Denote the path of this deletion $P = g_x, g_{v_2}, \ldots, g_{v_{n-1}}, g_y$. The corresponding path $P' = x, v_2, \ldots, v_{n-1}$ , $y$ is a Hamiltonian path in $G$. □

## 2.3 ILP formulation

We present an ILP formulation for the karyotype sorting problem. The formulation describes $d+1$ karyotype graphs $G^0, \ldots G^d$ corresponding to the genome after each operation. $G^0$ is set to $S$ and $G^d = T$. The formulation guarantees that difference between consecutive graphs corresponds to one valid operation of the model.

### *2.3.1* Variables

We define integer variables for each $G^k$, as follows. For every $k \in [0, d]$ and every $i \in \mathcal{G}$ let $\mathbf{cn}_i^k \in \mathbb{N}$ be the variable for the CN of gene $i$ after $k$ operations. By definition, $\text{cn}_i^0 = s\_\text{cn}_i$ and $\text{cn}_i^d = t\_\text{cn}_i$ for every $i \in \mathcal{G}$. Similarly, for every $k \in [0, d]$ and every $u, v \in \mathcal{E}$ let $\mathbf{adj}_{u,v}^k$ be the CN of a adjacency edge ($u$, $v$) after $k$ operations. By definition, $\text{adj}_{u,v}^0 = s\_\text{adj}_{u,v}$ and $\text{adj}_{u,v}^d = t\_\text{adj}_{u,v}$ for every $u, v \in \mathcal{E}$.

Now, we define binary variables for each type of operation. For every $k \in [0, d]$ and every $u, v \in \mathcal{E}$ let $\mathbf{cut}_{u,v}^k \in \{0, 1\}$ be an indicator variable for cutting the interval adjacency between $u$ and $v$ in the $k$'th operation. Similarly, $\mathbf{join}_{u,v}^k$ is an indicator variable for joining the two extremities $u$ and $v$ in the $k$'th operation. By convention, the cut and join are not symmetric, in order to support cutting or joining the same adjacency twice. An SCJ is a DCJ with one existing adjacency and an implicit adjacency of telomeres. To support SCJ operations, binary variables $\mathbf{addTel}_{t_1,t_2}^k$ are introduced for every two telomeres $t_1 \le t_2 \in \mathcal{T}$. $\text{addTel}_{t_1,t_2}^k = 1$ means that new copies of telomeres $t_1, t_2 \in \mathcal{T}$ are created.

For every $k \in [0, d]$ and every $u \le v \in \mathcal{E}^*$ let $\mathbf{ampAnch}_{u,v}^k \in \{0, 1\}$ be an indicator variable for a tandem duplication starting at $u$ and ending at $v$ in the $k$'th operation. In addition, the variable $\mathbf{ampAnch}_{t_1,t_2}^k \in \{0, 1\}$ for $t_1 \le t_2 \in \mathcal{T}$ indicates a chromosome amplification for the chromosome starting and ending at telomeres $t_1$ and $t_2$, respectively. Let $\mathbf{ampGene}_i^k \in \{0, 1\}$ be an indicator variable that gene $i$ [i.e. the interval edge $(i^t, i^h)$] is a part of the duplicated segment, and let $\mathbf{ampAdj}_{u,v}^k \in \{0, 1\}$ be an indicator variable that the adjacency edge ($u$, $v$) is a part of the duplicated segment.

Similarly, for every $k \in [0, d]$ and every $u \le v \in \mathcal{E}$, $\mathbf{delAnch}_{u,v}^k \in \{0, 1\}$ is an indicator variable for a deletion starting at $u$ and ending at $v$ in the $k$'th operation. $\mathbf{delGene}_i^k \in \{0, 1\}$ is an indicator variable that gene $i$ is a part of the deleted segment, and $\mathbf{delAdj}_{u,v}^k \in \{0, 1\}$ is an indicator variable that the adjacency edge ($u$, $v$) is a part of the deleted segment.

### *2.3.2* Constraints

We now describe the ILP constraints for each stage $0 \le k \le d-1$. We will describe constraints for each type of operation and general constraints for updating the karyotype graph.

**Updating the karyotypes**: The CN of a non-telomeric gene $i \in \mathcal{G}^*$ is increased by amplifications and decreased by deletions:

$$\text{cn}_i^{k+1} = \text{cn}_i^k + \text{ampGene}_i^k - \text{delGene}_i^k$$

For telomeric gene $i \in \mathcal{T}_\mathcal{G}$ with a corresponding extremity $t \in \mathcal{T}$, the CN can increase if new copies of the telomere are introduced via SCJ. An SCJ can add either two copies of the same telomere or one copy of two telomeres:

$$\text{cn}_i^{k+1} = \text{cn}_i^k + \text{ampGene}_i^k - \text{delGene}_i^k + 2\text{addTel}_{t,t}^k + \sum_{t' \ne t \in \mathcal{T}} \text{addTel}_{t,t'}^k$$

Updating adjacency CNs for internal nodes $u \ne v \in \mathcal{E}^*$:

$$\begin{aligned} \text{adj}_{u,v}^{k+1} = {} & \text{adj}_{u,v}^k + \text{ampAnch}_{u,v}^k + \text{ampAdj}_{u,v}^k + \text{delAnch}_{u,v}^k \\ & -\text{delAdj}_{u,v}^k - \text{cut}_{u,v}^k - \text{cut}_{v,u}^k + \text{join}_{u,v}^k + \text{join}_{v,u}^k \qquad (1) \end{aligned}$$

In words, we increase the CN of the adjacency if it is either, the anchor of a duplication, along the duplication path, the anchor of a deletion or its ends are joined. The adjacency CN is decreased if it is along a deletion path or it is cut. The cut and join variable can decrease or increase the CN by at most 2 if the same adjacency is used twice in a DCJ.

Updating adjacency CN for loop edges ($u$, $u$) is similar to Equation (1), but uses only single $\text{cut}_{u,u}^k$, $\text{join}_{u,u}^k$ variables.

For telomere $t \in \mathcal{T}$ and internal node $v \in \mathcal{E}^*$ we update the CN as follows:

$$\begin{aligned} \text{adj}_{t,v}^{k+1} = {} & \text{adj}_{t,v}^k + \text{ampAdj}_{t,v}^k + \text{delAnch}_{t,v}^k - \text{delAdj}_{t,v}^k \\ & -\text{cut}_{t,v}^k + \text{join}_{t,v}^k \end{aligned}$$

In addition, we require that in each stage $k$ there will be at most one operation:

$$\begin{aligned} & \frac{1}{2}\sum_{u,v \in \mathcal{E}} \text{cut}_{u,v}^k + \sum_{u \le v \in \mathcal{E}^*} \text{ampAnch}_{u,v}^k + \sum_{u \le v \in \mathcal{E}} \text{delAnch}_{u,v}^k \\ & + \sum_{t_1 \le t_2 \in \mathcal{T}} \text{ampAnch}_{u,v}^k \le 1 \end{aligned}$$

**DCJs**: An adjacency cannot be cut more times than its CN. Therefore, for every $u \in \mathcal{E}, v \in \mathcal{E}$ and $u \ne v$:

$$\text{cut}_{u,v}^k + \text{cut}_{v,u}^k \le \text{adj}_{u,v}^k$$

Similarly, for adjacencies of the form ($u$, $u$): $\text{cut}_{u,u}^k \le \text{adj}_{u,u}^k$.

For each adjacency $u \in \mathcal{E}$, the number of cuts equals to the number of joins:

$$\sum_{v \in \mathcal{E}} \text{cut}_{u,v}^k + \sum_{v \in \mathcal{E}} \text{cut}_{v,u}^k = \sum_{v \in \mathcal{E}} \text{join}_{u,v}^k + \sum_{v \in \mathcal{E}} \text{join}_{v,u}^k$$

For every pair of telomeres $t_1 \leq t_2 \in \mathcal{T}$, SCJ introduces an explicit adjacency $(t_1, t_2)$ which is cut immediately as part of a DCJ:

$$\text{cut}^k_{t_1,t_2} = \text{addTel}^k_{t_1,t_2}$$

In addition, if $t_1 < t_2$, set $\text{cut}^k_{t_2,t_1} = 0$. We also restrict that at most one pair of telomere copies is introduced as an SCJ in every stage:

$$\sum_{t1 \leq t_2 \in \mathcal{T}} \text{addTel}^k_{t_1,t_2} \leq 1$$

**Amplifications**: A gene $i \in \mathcal{G}$ cannot be amplified if it has CN = 0:

$$\text{ampGene}^k_i \leq \text{cn}^k_i \tag{2}$$

Similarly, an adjacency $u, v \in \mathcal{E}$ can only be amplified if it has a positive CN:

$$\text{ampAdj}^k_{u,v} \leq \text{adj}^k_{u,v} \tag{3}$$

For every internal node $u \in \mathcal{E}^*$, its corresponding gene is amplified if and only if one of its adjacencies is amplified or it is an anchor of an amplification:

$$\text{ampGene}^k_{g(u)} = \sum_{v \in \mathcal{E}^*} \text{ampAnch}^k_{u,v} + \sum_{v \in \mathcal{E}} \text{ampAdj}^k_{u,v}.$$

A telomere $t \in \mathcal{T}$ can only be involved in whole chromosome duplications. Therefore, the telomere is amplified iff it is an anchor iff one of its adjacencies is amplified:

$$\text{ampGene}^k_{g(t)} = \sum_{t' \in \mathcal{T}} \text{ampAnch}^k_{t,t'} = \sum_{v \in \mathcal{E}^*} \text{ampAdj}^k_{t,v}$$

**Enforcing path connectivity:** One problem with this formulation is that in addition to the amplification path, we may get a collection of disjoint cycles composed of alternating interval and adjacency edges with their corresponding variables $\text{ampGene}^k_{g(u)}$, $\text{ampAdj}^k_{u,v}$ set to one. For example, consider $S = 1, 1, 2, 2$ and $T = 1, 1, 1, 2, 2, 2$. To get from $S$ to $T$ we need to do two tandem duplications of the genes 1 and 2. However, according to the current formulation, this can be done in one step by assigning $\text{ampGene}^0_1 = 1, \text{ampGene}^0_2 = 1, \text{ampAnch}^0_{1^t,1^h} = 1, \text{ampAdj}^0_{2^t,2^t} = 1$.

To force the alternating path of the amplification to be connected, we add flow-like constraints (Bruckner *et al.*, 2010). Suppose $q > r$ are the anchors of the amplification and denote $r$ as the *sink*. Each node along the path from $q$ to $r$ (excluding $r$) will be as a source of one unit of flow, and we require that all flow will eventually be drained at $r$. This enforces the connectivity of the path.

Let $-2(n+1) \leq f^k_{u,v} \leq 2(n+1)$ be an integer variable for the directed amount of flow from $u \in \mathcal{E}$ to $v \in \mathcal{E}$. Let $\text{ampNodes}^k = 2 \sum_{i \in \mathcal{G}} \text{ampGene}^k_i$ be an integer variable for the number nodes that are amplified along the path. We seek $\text{ampNodes}^k - 1$ source nodes, each providing one unit of flow, and one sink that drains the $\text{ampNodes}^k - 1$ units of flow. Let $\text{sink}^k_v = \sum_{u > v} \text{ampAnch}^k_{v,u}$ be a binary variable indicating that $v$ is a sink. We have the following constraints:

The flow is anti-symmetric: $f^k_{u,v} = -f^k_{v,u}$.

An edge can contain flow only if it is amplified: $f^k_{u,v} \leq 2(n+1)$ $\text{ampAdj}^k_{u,v}$ if $u$, $v$ are not from the same gene, and $f^k_{u,v} \leq 2(n+1)$ $\text{ampGene}^k_i$ if $u$, $v$ are nodes of gene $i$.

Production and conservation of flow in every node $u \in \mathcal{E}$:

$$\sum_v f^k_{u,v} = \text{ampGene}^k_{g(u)} - \text{ampNodes}^k \cdot \text{sink}^k_u$$

By this constraint, if $u$ is not part of an amplification path, we have $\sum_v f^k_{u,v} = 0$. If $u$ is part of the amplification path, but not the sink, we have $\sum_v f^k_{u,v} = 1$, i.e. $u$ adds one unit to the flow. If $u$ is part of the amplification path and the sink, we have $\sum_v f^k_{u,v} = 1 - \text{ampNodes}^k$ and it drains all the flow.

Since the term $\text{ampNodes}^k \cdot \text{sink}^k_u$ is not linear we introduce a new non-negative integer variable $\text{product}^k_u$ such that $\text{product}^k_u = \text{ampNodes}^k \cdot \text{sink}^k_u$ using the following constraints:

$$\text{product}^k_u \leq \text{ampNodes}^k$$

$$\text{product}^k_u \leq 2(n+1) \cdot \text{sink}^k_u$$

$$\text{ampNodes}^k - 2(n+1)\left(1 - \text{sink}^k_u\right) \leq \text{product}^k_u$$

If $\text{ampNodes}^k = 0$ or $\text{sink}^k_u = 0$ then the first two constraints force $\text{product}^k_u = 0$, otherwise $\text{product}^k_u \leq \text{ampNodes}^k$. If $\text{sink}^k_u = 1$, we have $\text{ampNodes}^k \leq \text{product}^k_u$ and therefore $\text{product}^k_u = \text{ampNodes}^k$.

**Deletions:** Genes or adjacencies cannot be deleted if they have a CN of zero. Therefore, we add constraint similar to 2 and 3 using $\text{delGene}^k_i$ and $\text{delAdj}^k_{u,v}$ instead.

For every internal node $u \in \mathcal{E}^*$, one of its adjacencies is deleted if and only if it is an anchor of a deletion or its gene is deleted:

$$\sum_{v \in \mathcal{E}} \text{delAdj}^k_{u,v} = \text{delGene}^k_{g(u)} + \sum_{v \in \mathcal{E}^*} \text{delAnch}^k_{u,v}$$

A telomere $t \in \mathcal{T}$ can be deleted only if it is part of a whole chromosome deletion:

$$\text{delGene}^k_{g(t)} = \sum_{t' \in \mathcal{T}} \text{delAnch}^k_{t,t'}$$

If a telomere $t \in \mathcal{T}$ is an anchor of a segmental deletion then one of its adjacencies must be deleted:

$$\sum_{v \in \mathcal{E}} \text{delAnch}^k_{t,v} = \sum_{v \in \mathcal{E}} \text{delAdj}^k_{t,v}$$

As for amplifications, we add flow constraints to guarantee that the deletion path is connected.

### *2.3.3* Objective function

Our goal is to minimize the number of amplifications, deletions and DCJs:

$$\min \sum_{1 \leq k \leq d} \left( \sum_{t_1 \leq t_2 \in \mathcal{T}} \text{ampAnch}^k_{t_1,t_2} + \sum_{u \leq v \in \mathcal{E}^*} \text{ampAnch}^k_{u,v} + \sum_{u \leq v \in \mathcal{E}} \text{delAnch}^k_{u,v} + \frac{1}{4} \sum_{u,v \in \mathcal{E}} \left[ \text{cut}^k_{u,v} + \text{join}^k_{u,v} \right] \right)$$

DCJs have two cuts and two joins and therefore contribute one to the objective function. The objective function can be modified to give different weights to different operations, and even to specific events, e.g. amplifying regions with oncogenes.

### *2.3.4* Complexity

Overall, the ILP formulation has $O(dn^2)$ variables and $O(dn^2)$ constraints. We can relax the integrality constraints for all non-binary variables since each of them is constrained to be a sum of binary variables.

Since we do not know the optimal value $d^*$ of $d$, we can perform either a binary or sequential search on $d$. If there is no feasible solution for some $d$, we increase $d$. If $d \geq d^*$, the ILP will find a solution with $d^*$ operations since it can always add stages with no operations in them.

Each DCJ introduces two new adjacencies, while segmental deletions and tandem duplications introduce one new adjacency. A trivial lower bound on $d$ is the number of breakpoints divided by two. A trivial upper bound is $d \leq \sum_{u \leq v \in \mathcal{E}} |s\_\mathrm{adj}_{u,v} - t\_\mathrm{adj}_{u,v}| + \sum_{i \in \mathcal{G}} |s\_\mathrm{cn}_i - t\_\mathrm{cn}_i|$. That is, changing the CN of each gene and each adjacency separately.

The ILP problem is NP-hard (Karp, 1972) and the runtime of ILP algorithms is not polynomially bounded. However, modern ILP solvers incorporate powerful heuristics and can handle many large-scale problems. ILP has been a powerful tool in formulating and solving other rearrangement models (Rahmann and Klau, 2006; Shao *et al.*, 2015; Shao and Moret, 2015).

## 3 Results

### 3.1 Simulations

To assess performance, we simulated tumor karyotypes and applied the algorithm to them. Here is an overview of the simulation: We start with a diploid karyotype $S'$ with two identical chromosomes $1, \ldots, m$, and perform $d$ operations to derive a tumor karyotype $T'$. We then compress maximal identical segments in $S'$ and $T'$ into singletons. The resulting shorter source and target karyotypes are used as input to the algorithm.

Initially, each chromosome is represented by a sequence of $m = 100$ atomic segments. We perform a series of operations on the karyotype by applying duplications (tandem or chromosomal), deletions (segmental or chromosomal) and DCJs (reversals or translocations). Whole chromosome events are given low probability (5% each), while all other types are chosen uniformly at random. The span of segmental deletions, duplications and inversions was chosen at random and was limited to 30 units in order to avoid rapid loss of the middle segments.

In order to decrease the size of the karyotypes, we compress maximal identical sequences in $S'$ and $T'$. That is, a simple path that appears in $S'$ and $T'$ is compressed if all interval and adjacency edges along the path have the same CN, and nodes along the path have no other branching edges beside the path edges. The result is new karyotypes $S$ and $T$ with $n \leq m$ segments. This way, every segment in the compressed karyotypes must be involved in at least one breakpoint. Since all operations act on contiguous paths in the graph and all segments inside a compressed path are symmetric, we conjecture this procedure preserves the optimal distance. This compressed karyotype structure conforms with information provided by most assembly tools in which contiguous segments are determined by detecting breakpoints.

We simulated karyotypes with three to eight operations. Eitan and Shamir (2017) observed based on the analysis of tumor samples from Malhotra *et al.* (2013) that the average number of operations observed in real deep sequencing cancer data were 5–8 per connected component. For each distance, 20 instances were simulated and the optimal distance was computed by the algorithm. In Table 1, we see that the computed distance is bounded by the simulated distance but can sometimes be shorter when $d$ increases.

To test if the scenarios inferred are close to the simulated ones, we performed two additional comparisons, in terms of the types of operations and in terms of the actual operations. The results (Supplementary Figs S1 and S2) show that the scenarios are quite similar. We also observed that the distance from the karyotype back to the diploid genome is usually lower than the distance from the diploid to the karyotype (Supplementary Fig. S3). This is because one can use a few deletions to get rid of a mutated chromosome, and then create another copy of a normal chromosome in one operation.

**Table 1.** The optimal number of events computed by the algorithm versus the simulated number of events

| Simulated events | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Max | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Median | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 7 |
| Min | 1 | 1 | 1 | 2 | 1 | 2 | 3 | 6 |

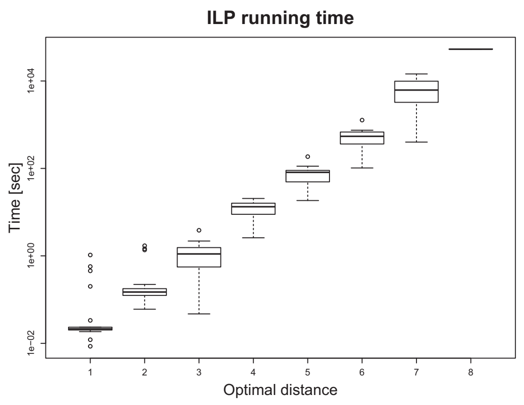

**Fig. 2.** ILP running time as a function of the optimal distance on simulated instances

Figure 2 shows the running time of the ILP algorithm as a function of the optimal distance it calculated. The time grows exponentially with the distance. The ILP was solved using Gurobi Optimizer 7.5 (Gurobi optimizer reference manual, Gurobi Optimization, 2018) on a shared Unix server with 72 2.3 GHz cores and 800 Gb of RAM.

### 3.2 Cancer karyotypes

We analyzed karyotypes from five ovarian cancer genomes that were sequenced as part of *TCGA* (Bell *et al.*, 2011) and were used in the analysis of PREGO (Oesper *et al.*, 2012). PREGO outputs CN per segment as well as for adjacencies based on the read coverage.

For each autosome in the genome, two telomeres were connected to the tail of the first segment and the head of the last segment with their CNs matching these segments (Supplementary Fig. S7). In each sample, we analyzed each connected component in the karyotype graph separately. For each such connected component, we calculated the distance from a matching diploid genome with the same subset of chromosomes. An example of the sequence of operations transforming OV4 chromosome 8 is shown in Figure 3.

To speed up the algorithm on real data we used several pre-processing steps. First, simple tandem duplications were removed from the karyotype and added to the distance. That is, for each $g \in \mathcal{G}$, we remove $\mathrm{adj}_{g^t,g^h}$ edges of the form $(g^t, g^h)$ from the graph as they can only be a result of tandem duplications. We again compress the karyotypes after this step.

In addition, some chromosome components exhibit large repetitions of complex chromosomal structures that are not simple tandem duplications. In such cases, we search for the longest path from one telomere to another that repeats itself $k \geq 2$ times. Such a path corresponds to an amplified chromosomal structure and thus we remove $k - 1$ repetitions from the karyotype. We use the algorithm to calculate the edit distance of this path separately and add it $k-1$ times to the total distance of the reduced graph.

We observed several examples of balanced (Supplementary Fig. S5) and unbalanced (Supplementary Fig. S4) translocations and also provide possible scenarios causing these phenomena. We also observed a breakage/fusion/bridge (*BFB*) cycle in chromosome 18 of OV2 (Supplementary Fig. S6). BFB cycles are a known source of genome instability (Greenman *et al.*, 2012). This aberrant chromosome is further amplified seven times, and is part of a complex connected component with chromosomes 12 and 16 (Supplementary Fig. S7). Similar observations were also shown by Oesper *et al.* (2012) without addressing the operation sequence.

Table 2 shows the distance calculated by the algorithm for each non-trivial connected component of the ovarian samples. The running time on the real karyotypes per connected component ranged from a few seconds for a simple component of distance at most four, to a few hours for more complex components. In three cases, the algorithm did not find any feasible solution within 24 h. These cases have very high CN or complex structural variations. All cases involve six or more interconnected chromosomes and contain interval CNs as high as 30.

## 4 Conclusions

In this study, we present a model for sorting a karyotype using deletions, amplifications, translocations and reversals. This model supports both structural and numerical alternations observed in cancer genomes. It focuses on finding a sequence of operations between two karyotypes and allows breakpoint reuse. We show the sorting problem is NP-hard and devise an ILP formulation that can find a shortest sequence of events that transform a normal into a tumor genome. We apply the algorithm on simulated karyotypes as well as real data of ovarian cancer. The algorithm is able to solve most components of the real tumor genomes.

The algorithm has limited applicability on highly complex karyotypes. As shown on simulated data (Fig. 2), running time grows exponentially with the number of operations. Additional work on the ILP formulation may make the approach more practical. On the real karyotype data, the algorithm could not resolve a few extremely rearranged connected components of chromosomes. Nevertheless, typical cancer samples exhibit modest complexity, making this algorithm useful in the majority of real cases (Eitan and Shamir, 2017). Moreover, a highly rearranged karyotype could be a result of noisy read data, tumor heterogeneity or unmodeled global events. Better methods are needed to address these cases.

While the model addresses a relatively wide array of operations, it still has some limitations. For instance, our duplication and deletion operations are restricted to simple paths with a single copy of each segment. However, in some scenarios we may benefit from performing operations on non-simple paths. For example, for a single segment with $m$ tandem repetitions, our model would require $m$ tandem duplications, but only $\log m$ operations will suffice if we allow non-simple path duplications. Other events like non-tandem segmental duplications and BFB (Zakov *et al.*, 2013) are not included (but are expressible, e.g. Supplementary Fig. S6) in the model.

Our karyotypes are represented using their CNs and adjacencies, but this representation is not unique for a specific set of chromosomes. That is, there could be several chromosome sets that may give the same karyotype. Since we do not model chromosomes explicitly, some operations may be artificial and would not correspond to operations on sequences.

In order to apply our method on more cancer data, we intend to improve the running time further. Then, a more systematic employment of the algorithm on a larger set of karyotypes can reveal sequences of operations common to several tumors. In addition, the algorithm can derive a sequence of operations between two tumor genomes (for example, from different time points) and thus help understand the evolution of tumors.

Ultimately, we would like to represent the chromosomes themselves and perform all operations on them. The goal in this case would be to decompose the source and target karyotypes into chromosomes such that the number of operations between them is minimum. Nonetheless, it was recently argued that re-construction of the exact cancer chromosomes remains a hard challenge (Eitan and Shamir, 2017).

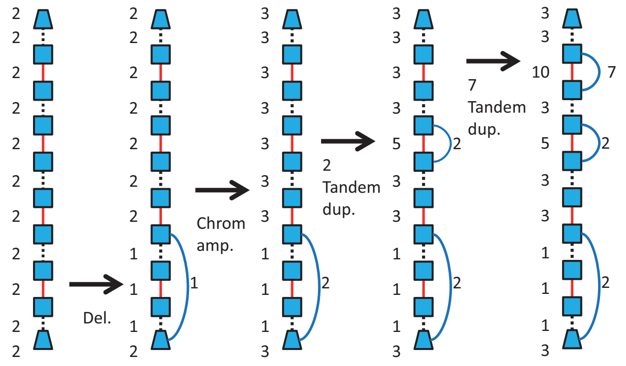

**Fig. 3.** Example of ovarian cancer sample OV4 chromosome 8 transformation from diploid (left) to tumor (right). Square nodes represent segment extremities and trapezoid nodes represent telomeres. Dotted edges correspond to adjacency edges, full straight edges correspond to interval edges and rounded edges correspond to novel adjacencies caused by the tumor process. The number next to each edge is its CN

**Table 2.** Results of the algorithm on TCGA ovarian samples

| Sample | Components | Distance | Sample | Components | Distance | Sample | Components | Distance |
|---|---|---|---|---|---|---|---|---|
| OV1 | (16),(13) | 1 | OV1 | (12, 15) | 3 | OV1 | (11, 20) | 5 |
| OV1 | (1, 2, 3, 4, 5, 6, 8, 9, 10, 14, 17, 19) | NA | OV2 | (8, 20) | 5 | OV2 | (3, 4),(9) | 6 |
| OV2 | (14, 21) | 10 | OV2 | (12, 16, 18) | 26 | OV2 | (1, 2, 5, 7, 10, 11, 15, 19, 22) | NA |
| OV3 | (13),(17),(21) | 1 | OV3 | (9),(18) | 2 | OV3 | (2) | 6 |
| OV3 | (4, 8) | 7 | OV4 | (1),(13),(20),(21) | 1 | OV4 | (18) | 2 |
| OV4 | (3),(15) | 6 | OV4 | (22) | 4 | OV4 | (11) | 10 |
| OV4 | (8),(9, 12) | 11 | OV4 | (5, 10, 16, 19) | 20 | OV4 | (2, 4, 6, 7, 14, 17) | NA |
| OV5 | (7),(16) | 1 | OV5 | (1, 3),(2, 17),(9, 10) | 3 | OV5 | (12, 21),(18) | 4 |

*Note*: The chromosomes involved in each component are shown within brackets. OV1: TCGA-13-0890; OV2: TCGA-13-0723; OV3: TCGA-24-0980; OV4: TCGA-24-1103 and OV5: TCGA-13-1411.

## Acknowledgement

The authors thank Layla Oesper for providing the pre-processed ovarian cancer data and Nimrod Rappoport for helpful comments.

## Funding

This study was supported in part by the Bella Walter Memorial Fund of the Israel Cancer Association and by Len Blavatnik and the Blavatnik Family foundation. R.Z. was supported by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University.

*Conflict of Interest*: none declared.

## References

Abo,R.P. (2015) BreaKmer: detection of structural variation in targeted massively parallel sequencing data using kmers. *NAR*, **43**, e19–e19.

Bader,M. (2010) Genome rearrangements with duplications. *BMC Bioinformatics*, **11**, S27.

Bafna,V. and Pevzner,P. (1996) Genome rearrangements and sorting by reversals. *SIAM J. Comput.*, **25**, 272–289.

Bell,D. *et al.* (2011) Integrated genomic analyses of ovarian carcinoma. *Nature*, **474**, 609–615.

Bergeron,A. *et al.* (2006) A unifying view of genome rearrangements. In: Bücher, P. and Moret, B.M. (eds.) *Algorithms in Bioinformatics*, Vol. **4175** of *LNCS*. Springer, Berlin, Heidelberg, pp. 163–173.

Bruckner,S. *et al.* (2010) Topology-free querying of protein interaction networks. *JCB*, **17**, 237–252.

Chowdhury,S.A. *et al.* (2014) Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLoS Comp. Bio.*, **10**, e1003740.

Ciriello,G. *et al.* (2013) Emerging landscape of oncogenic signatures across human cancers. *Nat. Genet.*, **45**, 1127–1133.

da Silva,P.H. *et al.* (2012) Restricted DCJ-indel model: sorting linear genomes with DCJ and indels. *BMC Bioinformatics*, **13**, S13.

Ding,L. *et al.* (2014) Expanding the computational toolbox for mining cancer genomes. *Nat. Rev. Genet.*, **15**, 556–570.

Eitan,R. and Shamir,R. (2017) Reconstructing cancer karyotypes from short read data: the half empty and half full glass. *BMC Bioinformatics*, **18**, 488.

Feijão,P. and Meidanis,J. (2011) SCJ: a breakpoint-like distance that simplifies several rearrangement problems. *TCBB*, **8**, 1318–1329.

Fielding,A.K. (2010) Current treatment of Philadelphia chromosome-positive acute lymphoblastic leukemia. *Haematologica*, **95**, 8–12.

Greenman,C.D. *et al.* (2012) Estimation of rearrangement phylogeny for cancer genomes. *Genome Res.*, **22**, 346–361.

Gurobi optimizer reference manual, Gurobi Optimization. (2018).

Hannenhalli,S. and Pevzner,P.A. (1996) Transforming men into mice (polynomial algorithm for genomic distance problem). In: *Proceedings of FOCS*, Vol. **36**, pp. 581–592. IEEE.

Hannenhalli,S. and Pevzner,P.A. (1999) Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, **46**, 1–27.

Karp,R.M. (1972) *Reducibility Among Combinatorial Problems*. Springer US, Boston, MA, pp. 85–103.

Korbel,J.O. *et al.* (2007) Paired-end mapping reveals extensive structural variation in the human genome. *Science*, **318**, 420–426.

Malhotra,A. *et al.* (2013) Breakpoint profiling of 64 cancer genomes reveals numerous complex rearrangements spawned by homology-independent mechanisms. *Genome Res.*, **23**, 762–776.

Ng,C.K. *et al.* (2012) The role of tandem duplicator phenotype in tumour evolution in high-grade serous ovarian cancer. *J. Pathol.*, **226**, 703–712.

Oesper,L. *et al.* (2012) Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinformatics*, **13(Suppl. 6)**, S10.

Ozery-Flato,M. and Shamir,R. (2009) Sorting cancer karyotypes by elementary operations. *JCB*, **16**, 1445–1460.

Pevzner,P. and Tesler,G. (2003) Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *PNAS*, **100**, 7672–7677.

Plesńik,J. (1979) The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Inf. Process. Lett.*, **8**, 199–201.

Rahmann,S. and Klau,G.W. (2006) Integer linear programs for discovering approximate gene clusters. In: *Proceedings of WABI*, Vol. **4175** of *LNCS*. Springer, Berlin Heidelberg, pp. 298–309.

Raphael,B.J. *et al.* (2003) Reconstructing tumor genome architectures. *Bioinformatics*, **19**, ii162–ii171.

Schwarz,R.F. *et al.* (2014) Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Comp. Bio.*, **10**, e1003535.

Shao,M. and Lin,Y. (2012) Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. *BMC Bioinformatics*, **13**, S13.

Shao,M. and Moret,B.M.E. (2015) Comparing genomes with rearrangements and segmental duplications. *Bioinformatics*, **31**, i329–i338.

Shao,M. *et al.* (2015) An exact algorithm to compute the double-cut-and-join jistance for jenomes with duplicate genes. *JCB*, **22**, 425–435.

Tannier,E. *et al.* (2009) Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, **10**, 120.

Vogelstein,B. *et al.* (2013) Cancer genome landscapes. *Science*, **339**, 1546–1558.

Yancopoulos,S. *et al.* (2005) Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, **21**, 3340–3346.

Zakov,S. *et al.* (2013) An algorithmic approach for breakage-fusion-bridge detection in tumor genomes. *PNAS*, **110**, 5546–5551.

Zeira,R. and Shamir,R. (2017) Sorting by cuts, joins, and whole chromosome duplications. *JCB*, **24**, 127–137.

Zeira,R. *et al.* (2017) A linear-time algorithm for the copy number transformation problem. *JCB*, **24**, 1179–1194.

# Sorting cancer karyotypes using DCJs, duplications and deletions

Ron Zeira and Ron Shamir

Supplementary information

## 1 Additional analysis of the simulation results

To test if the operations inferred by the ILP solutions are close to the simulated ones we compared them in two ways.

(1) We compared the difference in *types of operations*. We count the number of operations of each type (segmental deletion, chromosome deletion, segmental amplification, chromosome amplification and DCJ) in the simulated and inferred scenarios, and sum their absolute difference. The resulting value is $OpDiff = \sum_{X \in Ops} |\hat{X} - X|$, where $X$ and $\hat{X}$ are the number of operations in the simulated and the optimal sorting scenario and *ops* is the possible operation set. We call this value the *operation difference*. Figure S1 presents the operation difference of the simulated solution vs. the inferred solution.

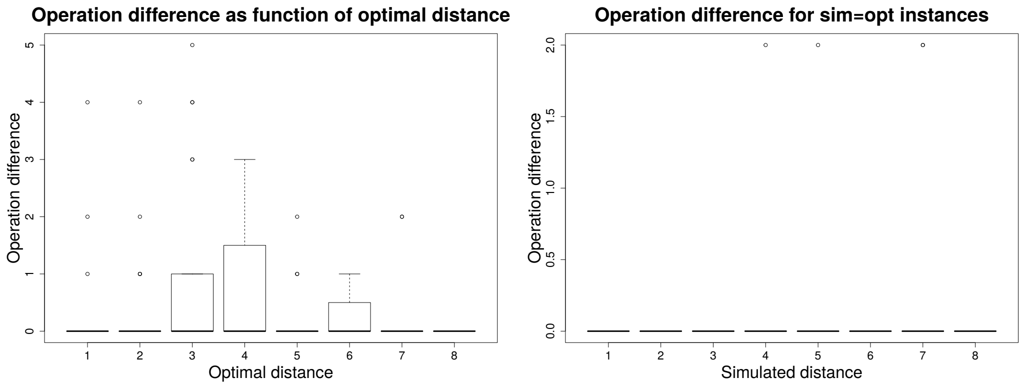

Figure S1: Operation difference. Left: difference as a function of the optimal distance. Right: difference as a function of the simulated distance, computed only for instances where the simulated distance equals the optimal distance.

(2) We compared the *actual operations* found by the optimal sorting scenario and in the simulated scenario. We define two operations to be the *identical* if they are of the same type and affect the same segments or adjacencies. They are *similar* if their segments or adjacencies are overlapping. In figure S2 we plot the number of identical and similar operations.

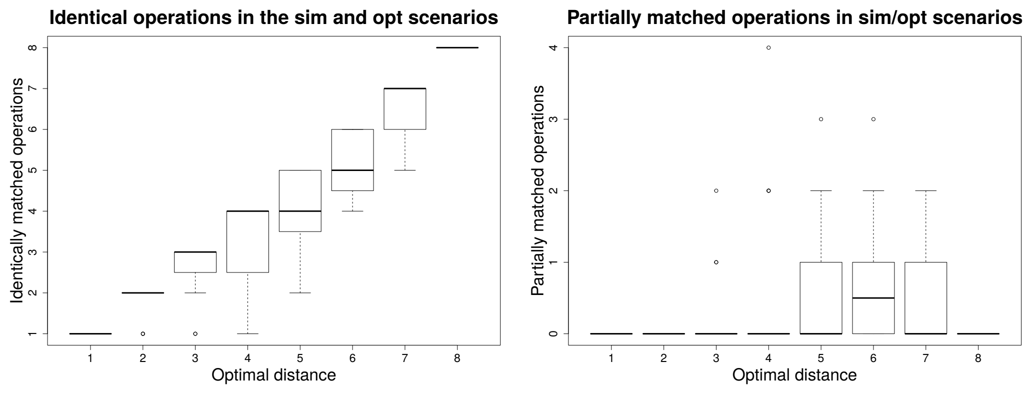

Figure S2: Comparing the operations in the simulated and inferred scenarios. Left: The number of operations that are identical in the optimal sorting scenario and the simulated scenario. Right: The number of operations that are similar in the optimal sorting scenario and the simulated scenario.

Both types of analysis show that the inferred operations are close to the simulated ones.

The distance between two karyotypes is not symmetric. For each simulated instance, we computed the optimal distance from the diploid karyotype $D$ to the simulated karyotype $K$, and the distance from $K$ to $D$. We call these the *forward* and *reverse* distance, respectively. Figure S3 plots the two distances. We see that the forward distance usually larger than the backward distance. The reason is that for a chromosome with multiple forward events, the backward scenario can delete or undo all events, and then add another copy of a normal chromosome to obtain a diploid chromosome.

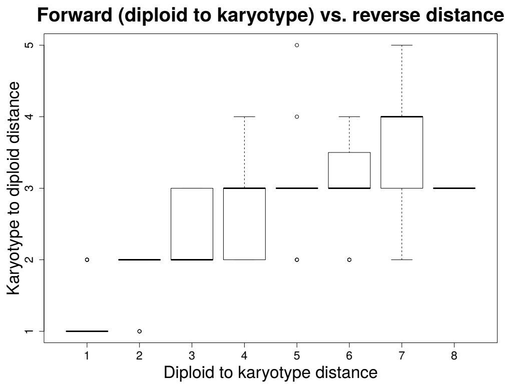

Figure S3: Comparison of the forward and reverse distance between the diploid and simulated karyotype. The figure shows only instances where both directions are feasible.

## 2 Additional tumor figures

The figures below show the inferred solutions for several TCGA ovarian cancer patient samples.

Figures S4 and S5 include examples of balanced and unbalanced translocations. In a *balanced translocation* two chromosomes exchange end segments, resulting in two mixed chromosome. In an *unbalanced translocation* a pair of the exchanged segments a missing, resulting with only one mixed chromosome.

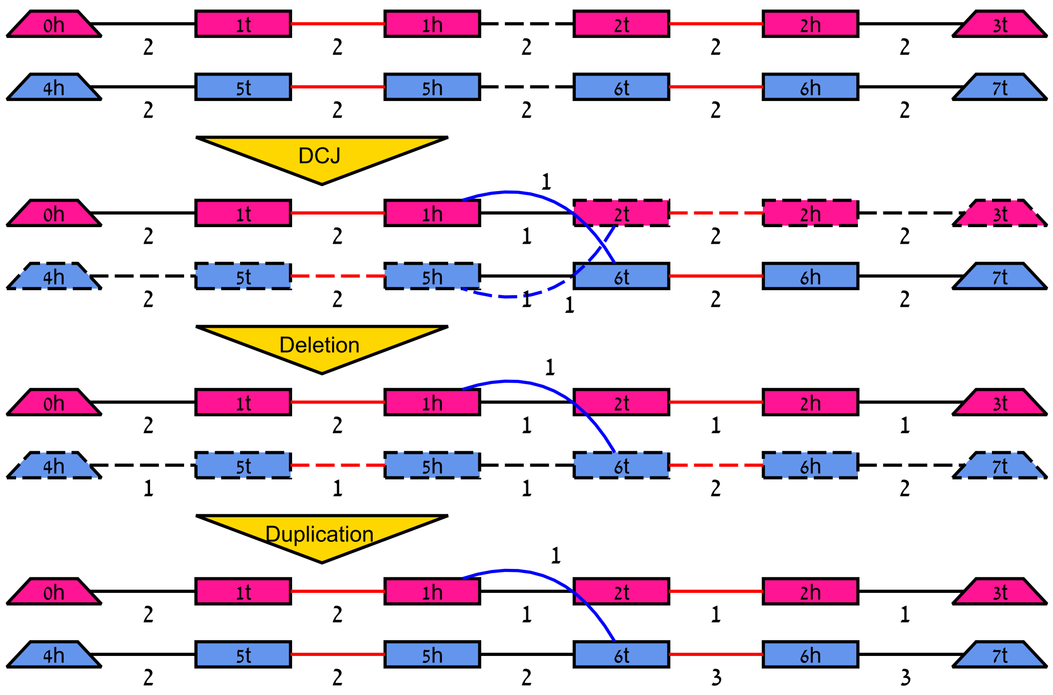

Figure S4: **Unbalanced translocation for sample OV5**. A sequence of inferred operations transforming chromosomes 2 (pink-upper) and 17 (teal-lower) from a diploid genome (top) to tumor genome (bottom). One operation (yellow triangle) transforms each genome into the next genome. The operation type is listed in the yellow triangle and the affected genes or adjacencies are dashed in each predecessor genome.

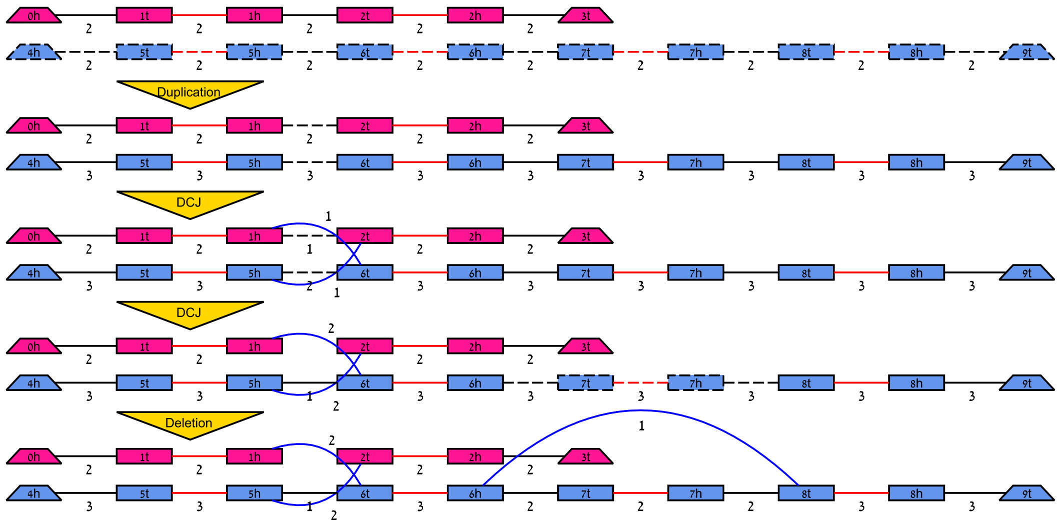

Figure S5: **Balanced translocation**. A sequence operations transforming chromosomes 11 (pink-upper) and 20 (teal-lower) from a diploid genome (top) to tumor genome (bottom) in sample OV1.

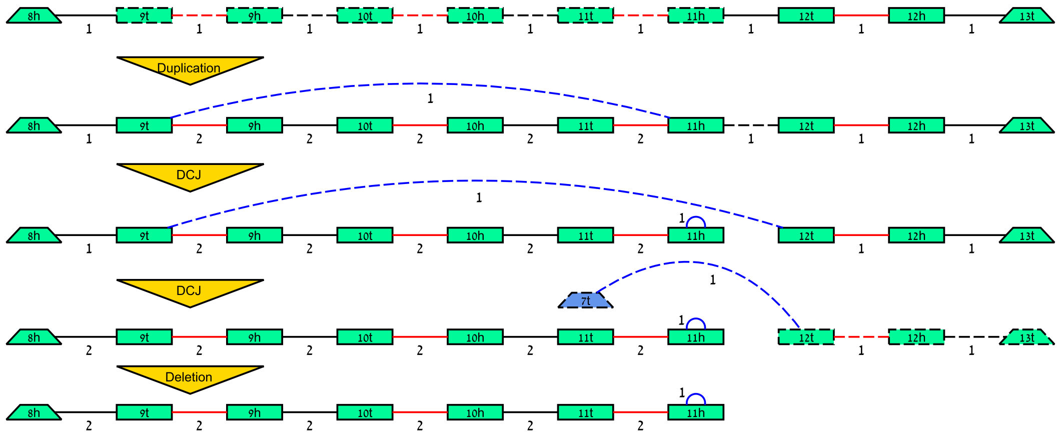

Figure S6: **BFB cycle**. An example of a Breakage/Fusion/Bridge Cycle in chromosome 18 of OV2. A normal chromosome 18 (top) is transformed into a BFB mutated genome (bottom) via a sequence of 4 operations. This mutated chromosome is further amplified 7 more times in OV2.

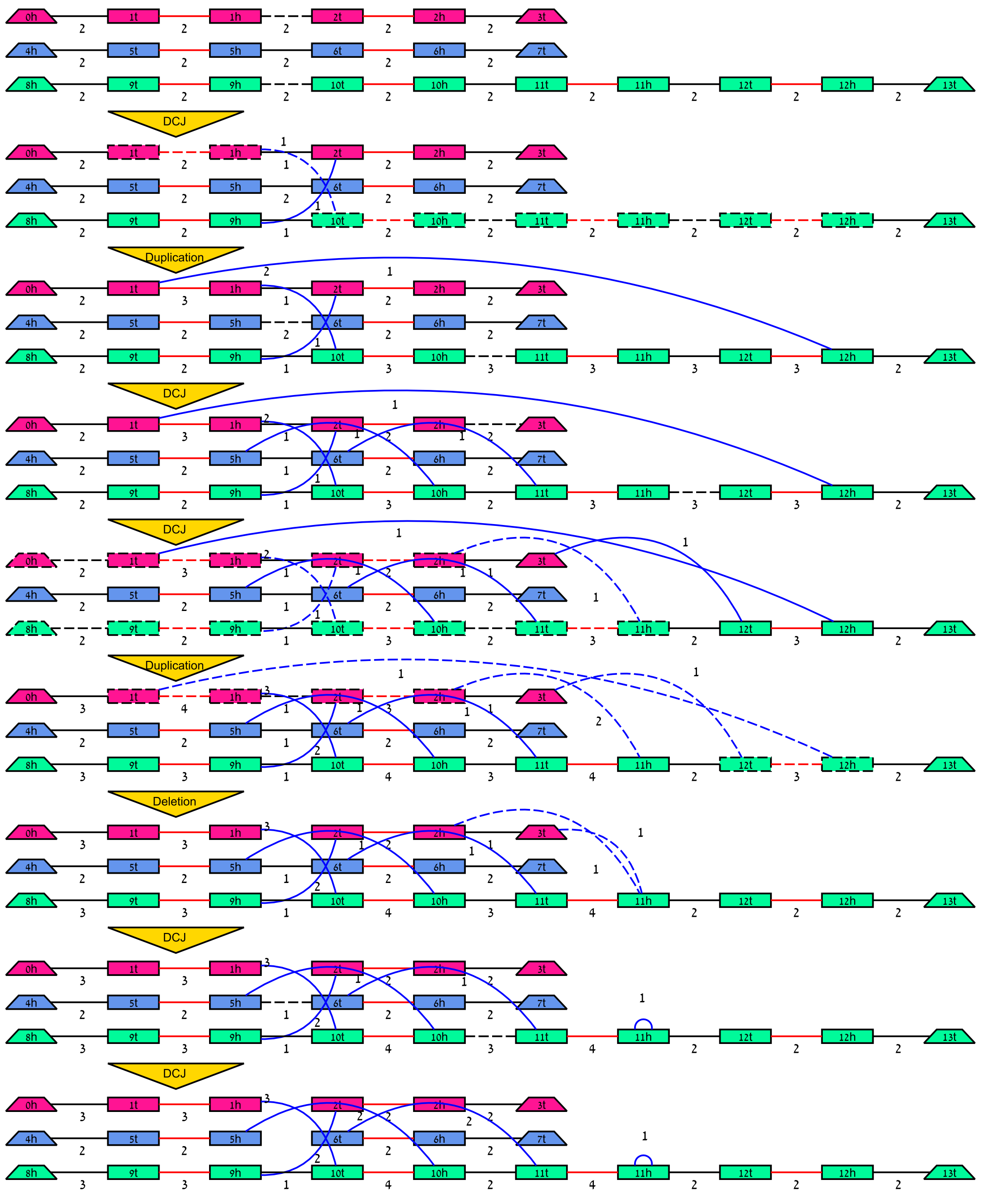

Figure S7: **Complex connected component**. A connected component of chromosome 12 (pink-upper), 16 (teal-middle) and 18 (green-lower) in sample OV2. Chromosome 18 is also a part of an amplified BFB cycle (figure S6).

# Chapter 5

# Discussion

In this thesis we developed computational models for structural and numerical aberrations occurring in cancer genomes. We presented three computational models of genome evolution by rearrangements designed for the analysis of cancer genomes. The first and simplest model accounted for breaking, joining and duplicating linear chromosomes. We gave a linear time algorithm for the sorting problem while showing a hardness result of a specific case. In the second model, we analyzed the transformation of vectors holding the number of copies each gene has in the genome via amplifications and deletions of contiguous segments. We showed linear, pseudo-polynomial and integer linear programming algorithms for several sorting problems under this model. Our third and most general model handled both structural and numerical rearrangements. We showed that the underlying sorting problem is computationally hard, gave an integer linear programming formulation to it and applied it on real samples from ovarian cancer.

In this chapter, we first summarize the models described in this thesis before characterizing possible extensions of them. Then, we outline a few possible directions for future research in cancer rearrangement models.

## 5.1 Sorting by cuts, joins and chromosome duplications

Chapter 2 presented the SCJD model, in which each operation may be either a breakage of a chromosome, a joining of two chromosomes or a duplication of a linear chromosome. The SCJD sorting problem seeks to find a shortest series of SCJD operations that transforms a given source genome with a single copy of each gene into another given target genome with exactly two copies for each gene. The SCJD distance is the length of such a sorting scenario.

We first showed that there exists an optimal SCJD sorting scenario where all duplication events are consecutive and separated from cuts and joins. Using this property, we gave a an optimization formula for the SCJD distance, and showed how to solve it optimally. The result is a linear time algorithm for the SCJD distance in which all intermediate genomes are linear. An optimal SCJD scenario that contains fewer duplication events can be viewed as more conservative. The assumption behind this is that duplications are more "radical" events than breakage or fusion, which are local events. We showed that the optimal scenario given by the algorithm performs a maximum number of duplications. In contrast, we also showed that finding an optimal scenario with a minimum number of duplications is NP-hard.

The SCJD model assumes that the source genome is ordinary and the target genome has two copies of every gene. The assumption on the target genome can be alleviated to at most two copies of each gene. But an extension required for it to be applicable to real data is allowing an arbitrary number of copies for each gene. Incorporating deletions into the model can make it more appropriate for cancer analysis. Finally, we might want to weight different operations according to their probability in order to reflect biology better. Nevertheless, we suspect a weight version could be NP-hard since we showed that finding an optimal sorting scenario with minimum duplications is hard.

## 5.2 Copy number transformation problems

In Chapter 3 we discussed a rearrangement model for the evolution of CNPs. The CNP of a tumor is an important tool in its analysis. Such profiles can be obtained by CGH arrays and deep sequencing. Schwarz *et al.* [94] define the edit distance between two copy number profiles as the number of segmental amplifications or deletions requires to transform one profile into the other. Nevertheless, the solution suggested in [94] is quite complex and its complexity appears to be exponential.

In this study we focused on the following fundamental problem: Given two CNPs, $u$ and $v$, compute the edit distance from $u$ to $v$, where the edit operations are segmental deletions and amplifications. We first showed the existence of an optimal sorting scenario with special properties such that all deletions precede all amplifications. These properties were then used to derive a pseudo-polynomial dynamic programming algorithm. Further analysis showed that partial solutions of this problem can be modeled with a piecewise linear function with three segments. We showed that the parameters of this function can be calculated in constant time and space, thus admitting a linear time, constant space algorithm for the CN transformation problem.

In a follow-up paper [36], in collaboration with Mohammed El-Kebir, Benjamin J. Raphael, Roded Sharan, Simone Zaccaria and Meirav Zehavi, we considered two extensions of this problem. Given two profiles, our first problem aims to find a parental profile that minimizes the sum of distances to the two profiles as its children. Given k profiles, the second, more general problem, seeks a phylogenetic tree, whose $k$ leaves are labeled by the given profiles and whose internal vertices are labeled by ancestral profiles such that the sum of edge distances is minimum. For the former problem we gave a pseudo-polynomial dynamic programming algorithm that is linear in the profile length, and an integer linear program formulation. For the latter problem we showed it is NP-hard and gave an integer linear program formulation. We assessed the efficiency and quality of our algorithms on simulated instances.

Several extensions of our model can be considered. Schwarz *et al.* [94] analyzed a model in which each CNP should be a sum of two CNPs corresponding to the maternal and paternal alleles and the goal is to minimize the transformation distance for these phased CNPs between the source and the target. The complexity aspects of the latter problem were not addressed and remain open. Giving different weight to amplifications and deletions, or even position dependent weighting is also possible. We conjecture that some of the properties we have shown on the optimal sorting scenario will not hold in the weighted case and additional analysis is needed in this case. Finally, experiments may give fractional copy numbers or have missing calls for certain positions due to noise and sample impurity. A recent paper tried to to decompose the clone mixture in order to reconstruct the evolutionary tree [121].

## 5.3 Sorting cancer karyotypes

Our final and most comprehensive model was presented in Chapter 4. We presented a model for the evolution of genomes with multiple gene copies using the operation types double-cut-and-joins, duplications and deletions. We studied the problem of finding a series of operations of minimum length that transforms one input karyotype graph into another.

We first showed that the problem is NP-hard and gave an integer linear programming formulation that solves the problem exactly under some mild assumptions. We tested our method on simulated genomes, showing it correctly detects the rearrangement operations and distance. We additionally applied our method on ovarian cancer genomes, displaying complex series of events.

Our method has a few limitations. First, the running time can become an issue for highly rearranged genomes, though we show its applicability to genomes of typical complexity. Proving additional properties on this model, such as restriction on the order of events or affected adjacencies, can be incorporated into the ILP and improve its run time. Second, the model has some assumptions on the operations affecting the genome. Third, the model uses an unlabeled graph representation of the genome and thus some operations may be artificial and would not correspond to operations on sequences. Further research on better modeling the genome and operations is needed.

One relatively simple extension of the model is adding weights to different operations. This would give the model a probabilistic interpretation and a shortest scenario would correspond to the most likely scenario. Such changes can be directly incorporated into the current ILP formulation. Another assumption in this model is that there is no noise in the source and target graphs. However, one might extend this model to allow noisy representation such as false adjacencies or fractional CNs. Finally, another possible expansion of this model can be phasing the parental and maternal alleles. Currently, the chromosomes are unlabeled and alleles are collapsed. We can modify our method to separate the source and target karyotypes into two graphs such that the distance between the pairs of phased genome is minimized. Alternatively, we can use as inputs phased or partially phased karyotype such as produced by [65, 85]

## 5.4 Future research in cancer rearrangements

Some cancer genomes were explained by complex structural and numerical events that are beyond the models discussed here. For example, a *breakage-fusion-bridge* (*BFB*) is an event in which a loss of a chromosome's end is followed by "doubling-up" and fusion of the surviving part (i.e., a chromosome $(a, b)$ is replaced by $(a, -a)$). In a BFB cycle, this process is repeated several times. Detection of BFB cycles can be done using sequencing and CN data [123, 122]. Dramatic rearrangement events also include *chromothripsis* and *chromoplexy*, in which one or more chromosomes are shattered into many pieces and some of the pieces are assembled in random order. Identifying these events in cancer genomes from sequencing data is still a hard challenge [74]. Computational models are in need to account for such events in the analysis of cancer evolution. Nevertheless, these events can sometimes be modeled by a series of simpler operations.

Advanced sequencing technologies could help in tackling GR problems in cancer. Long read sequencing techniques such as those of Pacific Biosciences and Oxford Nanopore can link distant DNA segments providing additional information on the relative location of different copies and simplify breakpoint identification [86, 54]. The linked short reads sequencing technology of 10X Genomics was recently shown to help in identifying structural variations in cancer genomes [40]. In addition, combining several sequencing technologies and sampling strategies together may improve rearrangement detection. A recent method by Dixon *et al.* [33] integrated optical mapping, high-throughput chromosome conformation capture (Hi-C), and whole-genome sequencing to accurately detect SVs in cancer genomes. We expect these technologies and others to play a prominent role in GR analysis in cancer in the years to come.

Single-cell sequencing technologies open new opportunities and challenges in computational cancer analysis [113]. Specifically, variations between individually sequenced cells taken from a tumor have been used to identify its evolutionary history [53, 64]. Detection of SVs and CNAs in single-cell sequencing is still a tough challenge due to the noise and biases in the data [111, 44]. The use of single-cell SVs or CNAs for clonal reconstruction has not been addressed yet, to the best of our knowledge. Additionally, one might use the heterogeneity among cells and their abundance in order to guide the rearrangement scenario. Alternatively, given a rearrangements scenario, we can try to map cells to stages along this sequence.

# Acronyms

aCGH - array comparative genomic hybridization

AG - adjacency graph

BFB - breakage-fusion-bridge

BG - breakpoint graph

BP - breakpoint

CGR - complex genomic rearrangement

CN - copy number

CNA - copy number alteration

CNO - copy number operation

CNP - copy number profile

DCJ - double cut and join

DNA - DeoxyriboNucleic Acid;

FISH - fluorescence in situ hybridization

GR - genome rearrangement

ILP - integer linear programming

SCoJ - single cut or join

SNV - single nucleotide variation

SV - structural variation

WGD - whole genome duplication

# Bibliography

[1] M. A. Alekseyev and P. A. Pevzner. Colored de Bruijn graphs and the genome halving problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):98–107, 2007.

[2] M. A. Alekseyev and P. A. Pevzner. Whole genome duplications and contracted breakpoint graphs. *SIAM Journal on Computing*, 36(6):1748–1763, 2007.

[3] S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes. In *Proc. RECOMB Comparative Genomics*, pages 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[4] S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. *Journal of Computational Biology*, 15(8):1093–115, 2008.

[5] P. Avdeyev, N. Alexeev, Y. Rong, and M. A. Alekseyev. A unified ILP framework for genome median, halving, and aliquoting problems under DCJ. In J. Meidanis and L. Nakhleh, editors, *Proc. RECOMB Comparative Genomics*, pages 156–178, Cham, 2017. Springer International Publishing.

[6] D. A. Bader, B. M. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[7] M. Bader. Sorting by reversals, block interchanges, tandem duplications, and deletions. *BMC Bioinformatics*, 10 Suppl 1:S9, 2009.

[8] M. Bader. Genome rearrangements with duplications. *BMC Bioinformatics*, 11 Suppl 1:S27, 2010.

[9] V. Bafna and P. Pevzner. Genome rearrangements and sorting by reversals.

*SIAM Journal on Computing*, 25(2):272–289, 1996.

[10] E. Barillot, L. Calzone, P. Hupé, J.-P. Vert, and A. Zinovyev. *Computational systems biology of cancer*. CRC Press, 2012.

[11] N. Beerenwinkel, C. D. Greenman, and J. Lagergren. Computational cancer biology: An evolutionary perspective. *PLOS Computational Biology*, 12(2):e1004717, 2016.

[12] A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.

[13] A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. In P. Bücher and B. M. Moret, editors, *Proc. Workshop on Algorithms in Bioinformatics*, volume 4175 of *LNCS*, pages 163–173. Springer, 2006.

[14] A. Bergeron, J. Mixtacki, and J. Stoye. On sorting by translocations. *Journal of Computational Biology*, 13(2):567–578, 2006.

[15] A. Bergeron, J. Mixtacki, and J. Stoye. A new linear time algorithm to compute the genomic distance via the double cut and join distance. *Theoretical Computer Science*, 410(51):5300–5316, 2009.

[16] P. Berman and S. Hannenhalli. Fast sorting by reversal. In *Proc. Combinatorial Pattern Matching*, pages 168–185. Springer, Berlin, Heidelberg, 1996.

[17] P. Biller, P. Feijão, and J. Meidanis. Rearrangement-based phylogeny using the Single-Cut-or-Join operation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(1):122–34, 2013.

[18] G. Blin, C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Comparing genomes with duplications: A computational complexity point of view. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(4):523–534, 2007.

[19] G. Blin, G. Fertin, and C. Chauve. The breakpoint distance for signed sequences. In *Proc. 1st Conference on Algorithms and Computational Methods for biochemical and Evolutionary Networks*, volume 3, pages 3–16. King's College London publications, 2004.

[20] J. E. Bowers, B. A. Chapman, J. Rong, and A. H. Paterson. Unravelling angiosperm genome evolution by phylogenetic analysis of chromosomal duplication events. *Nature*, 422(6930):433, 2003.

[21] M. D. V. Braga, R. Machado, L. C. Ribeiro, and J. Stoye. Genomic distance under gene substitutions. *BMC Bioinformatics*, 12 Suppl 9(Suppl 9):S8, 2011.

[22] M. D. V. Braga, E. Willing, and J. Stoye. Double cut and join with insertions

and deletions. *Journal of Computational Biology*, 18(9):1167–84, 2011.

[23] D. Bryant. The complexity of calculating exemplar distances. In D. Sankoff and J. H. Nadeau, editors, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families*, pages 207–211. Springer Netherlands, Dordrecht, 2000.

[24] A. Caprara. Sorting by reversals is difficult. In *Proc. First annual international conference on Research in Computational Molecular Biology*, pages 75–83, New York, New York, USA, 1997.

[25] X. Chen, J. Zheng, Z. Fu, et al. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–15, 2005.

[26] A. T. Chinwalla, L. L. Cook, K. D. Delehaunty, et al. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–562, 2002.

[27] S. A. Chowdhury, E. M. Gertz, D. Wangsa, et al. Inferring models of multiscale copy number evolution for single-tumor phylogenetics. *Bioinformatics*, 31(12):i258–67, 2015.

[28] S. A. Chowdhury, S. E. Shackney, K. Heselmeyer-Haddad, et al. Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLoS Computational Biology*, 10(7):e1003740, 2014.

[29] P. H. da Silva, R. Machado, S. Dantas, and M. D. V. Braga. Restricted DCJ-indel model: sorting linear genomes with DCJ and indels. *BMC Bioinformatics*, 13 Suppl 1(Suppl 19):S14, 2012.

[30] P. H. da Silva, R. Machado, S. Dantas, and M. D. V. Braga. DCJ-indel and DCJ-substitution distances with distinct operation costs. *Algorithms for Molecular Biology*, 8(1):21, 2013.

[31] L. Ding, T. J. Ley, D. E. Larson, et al. Clonal evolution in relapsed acute myeloid leukaemia revealed by whole-genome sequencing. *Nature*, 481(7382):506–10, 2012.

[32] L. Ding, M. C. Wendl, J. F. McMichael, and B. J. Raphael. Expanding the computational toolbox for mining cancer genomes. *Nature Reviews Genetics*, 15(8):556–570, 2014.

[33] J. R. Dixon, J. Xu, V. Dileep, et al. Integrative detection and analysis of structural variation in cancer genomes. *Nature Genetics*, page 1, 2018.

[34] M. Dzamba, A. K. Ramani, P. Buczkowicz, et al. Identification of com-

plex genomic rearrangements in cancers using CouGaR. *Genome Research*, 27(1):107–117, 2017.

[35] R. Eitan and R. Shamir. Reconstructing cancer karyotypes from short read data: the half empty and half full glass. *BMC Bioinformatics*, 18(1):488, 2017.

[36] M. El-Kebir, B. J. Raphael, R. Shamir, et al. Complexity and algorithms for copy-number evolution problems. *Algorithms for Molecular Biology*, 12(1):13, 2017.

[37] N. El-mabrouk. Sorting signed permutations by reversals and insertions/deletions of contiguous segments. *Journal of Discrete Algorithms*, 1(1):105–122, 2001.

[38] N. El-Mabrouk, J. H. Nadeau, and D. Sankoff. Genome halving. In M. Farach-Colton, editor, *Proc. Combinatorial Pattern Matching*, pages 235–250, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[39] N. El-Mabrouk and D. Sankoff. Analysis of gene order evolution beyond single-copy genes. *Methods in Molecular Biology*, 855:397–429, 2012.

[40] R. Elyanow, H.-T. Wu, and B. J. Raphael. Identifying structural variants using linked-read sequencing data. *Bioinformatics*, 34(2):353–360, 2018.

[41] P. Feijão and J. Meidanis. SCJ: a breakpoint-like distance that simplifies several rearrangement problems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(5):1318–29, 2011.

[42] G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements*. MIT Press, 2009.

[43] Z. Fu, X. Chen, V. Vacic, et al. MSOAR: A high-throughput ortholog assignment system based on genome rearrangement. *Journal of Computational Biology*, 14(9):1160–1175, 2007.

[44] T. Garvin, R. Aboukhalil, J. Kendall, et al. Interactive analysis and assessment of single-cell copy-number variations. *Nature Methods*, 12:1058 EP –, 2015.

[45] C. D. Greenman, E. D. Pleasance, S. Newman, et al. Estimation of rearrangement phylogeny for cancer genomes. *Genome Research*, 22(2):346–61, 2012.

[46] Y. Han. Improving the efficiency of sorting by reversals. In *Proc. 2006 International Conference on Bioinformatics and Computational Biology*, volume 6, pages 406–409. Citeseer, 2006.

[47] D. Hanahan and R. A. Weinberg. Hallmarks of cancer: the next generation. *Cell*, 144(5):646–74, 2011.

[48] S. Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71(1):137 – 151, 1996.

[49] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip. In *Proc. Annual ACM Symposium on the Theory of Computing*, volume 46, pages 178–189, New York, New York, USA, 1995.

[50] S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proc. IEEE Symposium on Foundations of Computer Science*, volume 36, pages 581–592, 1995.

[51] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, 1999.

[52] P. Hupé. Karyotype of the t47d breast cancer cell line. Wikimedia Commons file: https://commons.wikimedia.org/wiki/File:Karyotype_of_the_T47D_breast_cancer_cell_line.svg.

[53] K. Jahn, J. Kuipers, and N. Beerenwinkel. Tree inference for single-cell data. *Genome Biology*, 17(1):86, 2016.

[54] M. Jain, S. Koren, K. H. Miga, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology*, 36(4):338–345, 2018.

[55] G. Jean and M. Nikolski. Genome rearrangements: a correct algorithm for optimal capping. *Information Processing Letters*, 104(1):14 – 20, 2007.

[56] C. L. Kahn, B. H. Hristov, and B. J. Raphael. Parsimony and likelihood reconstruction of human segmental duplications. *Bioinformatics (Oxford, England)*, 26(18):i446–52, 2010.

[57] C. L. Kahn, S. Mozes, and B. J. Raphael. Efficient algorithms for analyzing segmental duplications with deletions and inversions in genomes. *Algorithms for Molecular Biology*, 5(1):11, 2010.

[58] C. L. Kahn and B. J. Raphael. Analysis of segmental duplications via duplication distance. *Bioinformatics (Oxford, England)*, 24(16):i133–8, 2008.

[59] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing*, 29(3):880, 1997.

[60] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13(1-2):180–210, 1995.

[61] J. D. Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Proc. Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 604–613, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[62] D. G. Kent and A. R. Green. Order Matters: The Order of Somatic Mutations Influences Cancer Evolution. *Cold Spring Harb Perspect Med*, 7(4), 2017.

[63] J. Kováč. On the complexity of rearrangement problems under the breakpoint distance. *Journal of Computational Biology*, 21(1):1–15, 2014.

[64] J. Kuipers, K. Jahn, B. J. Raphael, and N. Beerenwinkel. Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. *Genome Research*, 2017.

[65] Y. Li, S. Zhou, D. C. Schwartz, and J. Ma. Allele-specific quantification of structural variations in cancer genomes. *Cell Systems*, 3(1):21 – 34, 2016.

[66] H. Lodish, A. Berk, J. E. Darnell, et al. *Molecular Cell Biology*. Macmillan, 2008.

[67] A. McPherson, C. Wu, A. W. Wyatt, et al. nFuse: discovery of complex genomic rearrangements in cancer using high-throughput sequencing. *Genome Research*, 22(11):2250–61, 2012.

[68] F. Mitelman, B. Johansson, and F. Mertens. Mitelman database of chromosome aberrations and gene fusions in cancer, 2018. http://cgap.nci.nih.gov/Chromosomes/Mitelman.

[69] J. Mixtacki. Genome halving under DCJ revisited. In X. Hu and J. Wang, editors, *Proc. Computing and Combinatorics*, volume 5092 of *Lecture Notes in Computer Science*, pages 276–286. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[70] M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.

[71] J. H. Nadeau and B. A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences*, 81(3):814–818, 1984.

[72] C. K. Ng, S. L. Cooke, K. Howe, et al. The role of tandem duplicator phenotype in tumour evolution in high-grade serous ovarian cancer. *The Journal of Pathology*, 226(5):703–712, 2012.

[73] C. T. Nguyen, Y. C. Tay, and L. Zhang. Divide-and-conquer approach for the

exemplar breakpoint distance. *Bioinformatics*, 21(10):2171–2176, 2005.

[74] L. Oesper, S. Dantas, and B. J. Raphael. Identifying simultaneous rearrangements in cancer genomes. *Bioinformatics*, 34(2):346–352, 2018.

[75] L. Oesper, A. Ritz, S. J. Aerni, R. Drebin, and B. J. Raphael. Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinformatics*, 13 Suppl 6(Suppl 6):S10, 2012.

[76] M. Ozery-Flato and R. Shamir. Two notes on genome rearrangement. *Journal of Bioinformatics and Computational Biology*, 1(1):71–94, 2003.

[77] M. Ozery-Flato and R. Shamir. Sorting by translocations via reversals theory. In G. Bourque and N. El-Mabrouk, editors, *Proc. RECOMB Comparative Genomics*, pages 87–98, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[78] M. Ozery-Flato and R. Shamir. Sorting cancer karyotypes by elementary operations. *Journal of Computational Biology*, 16(10):1445–60, 2009.

[79] J. D. Palmer and L. A. Herbon. Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 28(1-2):87–97, 1988.

[80] B. Paten, D. R. Zerbino, G. Hickey, and D. Haussler. A unifying model of genome evolution under parsimony. *BMC Bioinformatics*, 15(1):206, 2014.

[81] P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes. *Genome Research*, 13(1):37–45, 2003.

[82] P. Pevzner and G. Tesler. Transforming men into mice. In *Proc. Seventh annual international conference on Research in Computational Molecular Biology*, pages 247–256, New York, New York, USA, 2003. ACM Press.

[83] D. Pinkel, T. Straume, and J. W. Gray. Cytogenetic analysis using quantitative, high-sensitivity, fluorescence hybridization. *Proceedings of the National Academy of Sciences*, 83(9):2934–2938, 1986.

[84] P. Popescu and H. Hayes. *Techniques in Animal Cytogenetics.* Springer Science & Business Media, 2000.

[85] A. Rajaraman and J. Ma. Toward Recovering Allele-specific Cancer Genome Graphs. *Journal of Computational Biology*, 25(7):624–636, 2018.

[86] A. Rhoads and K. F. Au. PacBio sequencing and its applications. *Genomics, Proteomics & Bioinformatics*, 13(5):278–289, 2015.

[87] D. P. Rubert, P. Feijão, M. D. V. Braga, J. Stoye, and F. H. V. Martinez. Approximating the DCJ distance of balanced genomes in linear time. *Algorithms*

*for Molecular Biology*, 12(1):3, 2017.

[88] J. Z. Sanborn, S. R. Salama, M. Grifford, et al. Double Minute Chromosomes in Glioblastoma Multiforme Are Revealed by Precise Reconstruction of Oncogenic Amplicons. *Cancer Research*, 73(19):6036–6045, 2013.

[89] D. Sankoff. Edit distance for genome comparison based on non-local operations. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. Combinatorial Pattern Matching*, pages 121–135, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[90] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.

[91] D. Sankoff, G. Leduc, N. Antoine, et al. Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome. *Proceedings of the National Academy of Sciences*, 89(14):6575–6579, 1992.

[92] H. Scherthan, T. Cremer, U. Arnason, et al. Comparative chromosome painting discloses homologous segments in distantly related mammals. *Nature Genetics*, 6(4):342, 1994.

[93] R. F. Schwarz, C. K. Y. Ng, S. L. Cooke, et al. Spatial and temporal heterogeneity in high-grade serous ovarian cancer: A phylogenetic analysis. *PLOS Medicine*, 12(2):e1001789, 2015.

[94] R. F. Schwarz, A. Trinh, B. Sipos, et al. Phylogenetic quantification of intratumour heterogeneity. *PLoS Computational Biology*, 10(4):e1003535, 2014.

[95] R. Shamir, M. Zehavi, and R. Zeira. A Linear-Time Algorithm for the Copy Number Transformation Problem. In *Proc. Combinatorial Pattern Matching*, volume 54 of *LIPIcs*, pages 16:1—-16:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[96] M. Shao and Y. Lin. Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. *BMC Bioinformatics*, 13(Suppl 19):S13, 2012.

[97] M. Shao, Y. Lin, and B. M. Moret. An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. *Journal of Computational Biology*, 22(5):425–435, 2015.

[98] M. Shao and B. M. Moret. A fast and exact algorithm for the exemplar breakpoint distance. *Journal of Computational Biology*, 23(5):337–346, 2016.

[99] M. Shao and B. M. Moret. On computing breakpoint distances for genomes with duplicate genes. *Journal of Computational Biology*, 24(6):571–580, 2017.

[100] M. Shao and B. M. E. Moret. Comparing genomes with rearrangements and segmental duplications. *Bioinformatics*, 31(12):i329–i338, 2015.

[101] G. Shi, L. Zhang, and T. Jiang. MSOAR 2.0: Incorporating tandem duplications into ortholog assignment based on genome rearrangement. *BMC Bioinformatics*, 11(1):10, 2010.

[102] S. H. Strauss, J. D. Palmer, G. T. Howe, and A. H. Doerksen. Chloroplast genomes of two conifers lack a large inverted repeat and are extensively rearranged. *Proceedings of the National Academy of Sciences*, 85(11):3898–3902, 1988.

[103] A. H. Sturtevant and T. Dobzhansky. Inversions in the third chromosome of wild races of drosophila pseudoobscura, and their use in the study of the history of the species. *Proceedings of the National Academy of Sciences*, 22(7):448–450, 1936.

[104] J. Suksawatchon, C. Lursinsap, and M. Bodén. Computing the reversal distance between genomes in the presence of multi-gene families via binary integer programming. *Journal of Bioinformatics and Computational Biology*, 5(1):117–33, 2007.

[105] E. Tannier, A. Bergeron, and M.-F. Sagot. Advances on sorting by reversals. *Discrete Applied Mathematics*, 155(6-7):881–888, 2007.

[106] E. Tannier, C. Zheng, and D. Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, 10(1):120, 2009.

[107] L. Tattini, R. D'Aurizio, and A. Magi. Detection of Genomic Structural Variants from Next-Generation Sequencing Data. *Frontiers in Bioengineering and Biotechnology*, 3:92, 2015.

[108] G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.*, 65(3):587–609, 2002.

[109] G. Tesler. GRIMM: genome rearrangements web server. *Bioinformatics*, 18(3):492–493, 2002.

[110] A. E. Urban, J. O. Korbel, R. Selzer, et al. High-resolution mapping of DNA copy alterations in human chromosome 22 using high-density tiling oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 103(12):4534–9, 2006.

[111] T. Voet, P. Kumar, P. Van Loo, et al. Single-cell paired-end genome sequencing reveals structural variation per cell cycle. *Nucleic Acids Research*, 41(12):6119–

6138, 2013.

[112] B. Vogelstein, N. Papadopoulos, V. E. Velculescu, et al. Cancer genome landscapes. *Science*, 339(6127):1546–58, 2013.

[113] Y. Wang, J. Waters, M. L. Leung, et al. Clonal evolution in breast cancer revealed by single nucleus genome sequencing. *Nature*, 512:155 EP –, 2014. Article.

[114] R. Warren and D. Sankoff. Genome halving with double cut and join. *Journal of Computational Biology*, 7(2):357–371, 2009.

[115] R. Warren and D. Sankoff. Genome aliquoting revisited. *Journal of Computational Biology*, 18(9):1065–1075, 2011.

[116] E. Willing, S. Zaccaria, M. D. Braga, and J. Stoye. On the inversion-indel distance. *BMC Bioinformatics*, 14 Suppl 15:S3, 2013.

[117] K. H. Wolfe and D. C. Shields. Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387:708 EP –, 1997.

[118] S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

[119] S. Yancopoulos and R. Friedberg. DCJ path formulation for genome transformations which include insertions, deletions, and duplications. *Journal of Computational Biology*, 16(10):1311–38, 2009.

[120] Z. Yin, J. Tang, S. W. Schaeffer, and D. A. Bader. Exemplar or matching: modeling DCJ problems with unequal content genome data. *Journal of Combinatorial Optimization*, 2015.

[121] S. Zaccaria, M. El-Kebir, G. W. Klau, and B. J. Raphael. Phylogenetic Copy-Number Factorization of Multiple Tumor Samples. *Journal of Computational Biology*, page cmb.2017.0253, 2018.

[122] S. Zakov and V. Bafna. Reconstructing Breakage Fusion Bridge Architectures Using Noisy Copy Numbers. *Journal of Computational Biology*, 22(6):577–594, 2015.

[123] S. Zakov, M. Kinsella, and V. Bafna. An algorithmic approach for breakage-fusion-bridge detection in tumor genomes. *Proceedings of the National Academy of Sciences*, 110(14):5546–51, 2013.

[124] R. Zeira and R. Shamir. Sorting by cuts, joins and whole chromosome duplications. In *Proc. Combinatorial Pattern Matching*, volume 9133 of *LNCS*, pages 396–409. Springer, 2015.

[125] R. Zeira and R. Shamir. Sorting by cuts, joins, and whole chromosome duplications. *Journal of Computational Biology*, 24(2):127–137, 2017.

[126] R. Zeira and R. Shamir. Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions. *Bioinformatics*, page bty381, 2018.

[127] R. Zeira, M. Zehavi, and R. Shamir. A linear-time algorithm for the copy number transformation problem. *Journal of Computational Biology*, 24(12):1179–1194, 2017.

[128] D. R. Zerbino, T. Ballinger, B. Paten, G. Hickey, and D. Haussler. Representing and decomposing genomic structural variants as balanced integer flows on sequence graphs. *BMC Bioinformatics*, 17(1):400, 2016.

[129] C. Zheng, Q. Zhu, and D. Sankoff. Genome halving with an outgroup. *Evolutionary Bioinformatics Online*, 2:295–302, 2007.

[130] Zimao Li, Lusheng Wang, and Kaizhong Zhang. Algorithmic approaches for genome rearrangement: a review. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 36(5):636–648, 2006.

3. **Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions**
   Ron Zeira and Ron Shamir
   Published in *Bioinformatics* [119]

**מוטיבציה**: בעיות רה-ארגון הגנום הן מרכזיות הן באבולוציה והן בסרטן. רוב המודלים לרה-ארגון הגנום מניחים שהגנום מכיל עותק בודד מכל גן ושהשינויים היחידים בגנום הם מבניים, זאת אומרת, שינוי של סדר המקטעים. לעומת זאת, גנומים סרטניים עוברים שינויים כמותיים כגון מחיקות והכפלות, ולכן מספר העותקים של כל גן משתנה. טיפול בתוכן לא שוויוני של גנים היא משימה מאתגרת שמעט אלגוריתמים התייחסו אליה. מודלים מציאותיים יותר נדרשים על מנת לסייע לעקוב אחר התפתחות הגנום בתהליך הסרטני.

**תוצאות**: כאן אנו מציגים מודל לאבולוציה של גנומים עם מספר עותקים מרובה בעזרת פעולות מסוג חיתוך-כפול-והדבקה, הכפלה ומחיקה. האירועים שנתמכים על ידי המודל הם *היפוכים, שחלופים, הכפלות עוקבות, מחיקות מקטעים והכפלות ומחיקות של כרומוזומים*, המכסים את מרבית סוגי השינויים המבניים והכמותיים שנצפים בדגימות סרטן. מטרתנו היא למצוא סדרת פעולות באורך קצר ביותר שממירה קריוטיפ אחד לאחר. אנו מראים שהבעיה היא קשה ונותנים ניסוח תכנות לינארי בשלמים שפותר את הבעיה במדויק תחת מספר הנחות קלות. אנו בודקים את השיטה שלנו בגנומים שיוצרו בסימולציה וכן על גנומים מסרטן השחלות. המחקר שלנו מקדם את המחקר העדכני ביותר בשתי צורות: הוא מתיר מגוון רחב יותר של פעולות מאשר המודלים הקיימים ולכן הוא מציאותי יותר, וכן הוא המחקר הראשון שמנסה לשחזר סדרה שלמה של פעולות מבניות וכמותיות באבולוציה של סרטן.

בנוסף לכן, פרק 1 מבוסס על מאמר הסקירה

"**Genome Rearrangement Problems with Single and Multiple Gene Copies: A Review**" by Ron Zeira and Ron Shamir

המאמר עבר שיפוט והתקבל לפרסום בספר "Festschrift in honor of Bernard M.E. Moret" בעריכת T. Warnow שאמור להתפרסם ע"י הוצאת Springer ב-2019.

# תקציר המאמרים הכלולים בתזה

להלן תקצירי המאמרים עליהם מבוססת עבודה זו :

1. **Sorting by cuts, joins and whole chromosome duplications**
   Ron Zeira and Ron Shamir
   Published in *Proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching (CPM 2015)* [117] and as a full version in *Journal of Computational Biology (JCB)* [118]

בעיות רה-ארגון הגנום נחקרו בצורה מקיפה עקב חשיבותן הרבה בביולוגיה. מרבית המודלים שנחקרו מניחים שלכל גן קיים עותק יחיד. אולם, במציאות גנים משוכפלים די נפוצים, במיוחד בסרטן. כאן אנו לוקחים צעד קדימה לקראת טיפול בגנים משוכפלים ע"י הצעת מודל שמתיר פעולות בסיסיות של חיתוך, חיבור והכפלה של כרומוזום שלם. בהינתן שני גנומים לינאריים **א** עם עותק אחד לכל גן ו-**ב** עם שני עותקים לכל גן, אנו נותנים אלגוריתם לינארי לחישוב סדרה קצרה ביותר של פעולות שממירות את **א** ל-**ב** כך שכל הגנומי הביניים הם לינארים. אנו גם מראים שחישוב סדרה אופטימלית עם מספר קטן ביותר של הכפלות היא בעיה קשה.

2. **A linear-time algorithm for the copy number transformation problem**
   Ron Zeira, Meirav Zehavi and Ron Shamir
   Published in *Proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)* [88] and as a full version in *Journal of Computational Biology (JCB)* [120]

בעיות רה-ארגון הגנום הן מרכזיות הן באבולוציה והן בסרטן. רוב התרחישים האבולוציוניים נחקרו תחת ההנחה שהגנום מכיל עותק יחדי מכל גן. לעומת זאת, גנומים סרטניים עוברים מחיקות והכפלות וכתוצאה מזה מספר העותקים של כל גן משתנה. מספר העותקים של כל מקטע לאורך הכרומוזום נקרא *פרופיל מספר העותקים* שלו. הבנת השינויים בפרופיל מספר העותקים יכולה לסייע בחיזוי התקדמות המחלה והטיפול בה. נכון להיום, שאלות הנוגעות למרחק בין פרופילי מספר עותקים זכו לתשומת לב מדעית מועטה. כאן אנו מתמקדים בבעיה הבסיסית הבאה שהוצגה על ידי שווארץ ושותפיו [87]: בהינתן שני פרופילים של מספר עותקים **א** ו-**ב**, חשב את המספר המינימלי של פעולות אשר ממירים את **א** ל-**ב**. אנו מיישבים את היעילות החישובית של בעיה זו ומראים שהיא פתירה בזמן לינארי ומקום קבוע.

## תוצאות

בתיזה זו אנו מפתחים מודלים לשינויים מבניים וכמותיים שקורים בגנומים סרטניים. אנו מציגים שלושה מודלים חישוביים להתפתחות הגנום על ידי ארגון מחדש שמיועדים לניתוח גנומים סרטניים.

פרק 2 מציג מודל הנקרא SCJD, ובו כל פעולה או חותכת כרומוזום לשניים, או מאחדת שני כרומוזומים או מכפילה כרומוזום שלם. בעיית הסידור שאנו חוקרים היא איך להפוך גנום ובו עותק יחיד מכל גן לגנום ובו שני עותקים מכל גן בעזרת מספר קטן ככל האפשר של פעולות. אנו מראים תחילה שישנו תרחיש של סידור כזה בו כל פעולות ההכפלה באות אחת אחרי השנייה. בעזרת תכונה זו אנו נותנים נוסחה למרחק בין הגנומים ונותנים אלגוריתם למציאת סדרת הפעולות הקצרה ביותר. ניתן לחשוב שסדרת פעולות כזו המכילה מספר הכפלות מועט היא יותר שמרנית מכיוון שפעולות ההכפלה הן יותר "קיצוניות" מאשר איחוד או חיתוך. אנו מראים שסדרת הפעולות שהאלגוריתם שלנו מוצא מכילה מספר מרבי של הכפלות ואילו למצוא סדרת פעולות קצרה ביותר עם המספר הקטן ביותר של הכפלות היא בעיה קשה חישובית.

בפרק 3 אנו דנים במודל של שינוי במספר העותקים בגנום. פרופיל מספר העותקים הוא וקטור שמתאר את המספר העותקים של כל גן בגנום. המטרה במודל זה היא להפוך פרופיל אחד למשנהו בעזרת מספר קטן ביותר של הכפלות ומחיקות. פעולות אלו מעלות או מורידות את מספר העותקים של גנים סמוכים, כאשר גן שמאבד את כל עותקיו לא מושפע יותר מפעולות אלו. אנו מראים תחילה שקיימת סדרת פעולות בין שני הפרופים בעלת תכונות מיוחדות ומשתמשים בהן לניסוח אלגוריתם תכנות דינמי. ניתוח מדויק יותר של האלגוריתם מאפשר לנו לשפר אותו כך שירוץ בזמן לינארי באורך הפרופילים וישתמש בכמות מקום קבועה.

המודל האחרון והמקיף ביותר שלנו מוצג בפרק 4. אנו מציגים בו מודל להתפתחות של גנומים עם מספר עותקים מרובה בעזרת פעולות הכפלה, מחיקה וחיתוך-כפול-והדבקה (double-cut-and-join). המטרה היא למצוא סדרת פעולות קצרה ביותר שהופכת קריוטיפ אחד לשני. אנו מוכיחים תחיל שהבעיה היא קשה חישובית, אולם נותנים פתרון מעשי לבעיה המבוסס על תכנות לינארי בשלמים. אנו מדמים קריוטיפים מתוך אבולוציה של סרטן ובודקים את האלגוריתם עליהם. בנוסף, אנו מפעילים את האלגוריתם על דגימות של סרטן השחלות ומזהים סדרות פעולות מורכבות בהן.

## רקע כללי

הגנום מקודד הוראות המשמשות בהתפתחות ובתפקוד של כל היצורים החיים. גנומים בנויים מ-ד.נ.א (DNA), מולקולה הבנויה מרצף של בסיסים, כאשר כל בסיס יכול להיות אחד מארבעה סוגים. הגנום הוא סך כל ה-ד.נ.א בתא והוא מופרד לתתי רצפים הנקראים כרומוזומים. גן הוא מקטע לאורך הכרומוזום המכיל את המידע הדרוש לבניית חלבון, אבן הבניין למרבית התפקוד התאי. גנים הם יחידת ההורשה הבסיסית המועברת מדור לדור. הגורמים למגוון היצורים החיים הם שינויים בגנום בין דורות. שינויים אלו נובעים מהעתקה לא מדויקת של החומר הגנטי וכן השפעות סביבתיות, והם פותחים הזדמנויות לגנים חדשים, משופרים ולבסוף למיני יצורים חדשים.

גנומים מתפתחים בתהליך של שינויים מקומיים וגלובליים. שינויים מקומיים מתייחסים לשינויים של בסיס בודד או מספר קטן של בסיסים. שינויים אלו מחליפים בסיס בבסיס, מוסיפים בסיס או מוחקים בסיס. לעומת זאת, רצף ה-ד.נ.א יכול להשתנות על ידי שינוי המבנה שלו בצורה רחבה יותר. השינויים הגלובליים, הנקראים גם ארגון מחדש של הגנום (genome rearrangements) או שינויים מבניים (structural variations), מזיזים, משכפלים או מוחקים מקטעים גדולים של ד.נ.א.

הגנומים של מינים קרובים הם דומים. ההבדל בין הגנומים נובע בעיקר משינויים מבניים שקרו לאחר הפיצול האבולוציוני בין המינים. שינויים אלו נחקרו במגוון מינים והם מעלים את השאלה איך שינויים אלו נוצרו, השאלה הבסיסית של תחום הארגון הגנומי. המחקר החישובי של ארגון הגנום נולד לפני כ-25 שנה. מאז הוא שגשג והתפתח לתחום מחקר מרתק, המשלב בין מודלים מתמטיים, תיאוריה אלגנטית ויישומים למידע ביולוגי. המודלים המוקדמים שהונעו על ידי אבולוציה של מינים היו פשוטים (למרות שהניתוח שלהם היה מתוחכם) והניחו שהגנום מכיל עותק בודד מכל גן. ככל שהמידע הביולוגי שנאסף גדל, הזדמנויות חדש לניתוח נוצרו, ויצרו דרישה למודלים מורכבים יותר ותיאוריה חדשה.

סרטן היא מחלה מורכבת שמונעת על ידי הצטברות של שינויים ב-ד.נ.א במהלך דורות של התחלקות התא. שינויים אלו משפיעים על התפתחות הגידול, התקדמות המחלה, "בריחה" ממערכת החיסון ועמידות לתרופות. גם בסרטן, חלק מהשגיאות הן מקומיות ומשנות בסיס בודד. אולם מספר השגיאות הללו יכול להצטבר לאלפים. מצד שני, שינויים גדולים שמשנים את ארגון הגנום יכולים להזיז מקטעי ד.נ.א. שגיאות שמשנות את כמות החומר הגנטי נקראות שינויים במספר העותקים (copy number alterations) ומכילות הכפלות ומחיקות של מקטעים גנומיים. הקריוטיפ של תא הוא קבוצת הכרומוזומים שלו, המכילה הן את הכמות והן את המבנה של הכרומוזומים. שגיאות מבניות גדולות יכולות להשפיע בצורה משמעותית על הקריוטיפ של התא. שגיאות אלו יכולות להגביר גנים שמעודדים סרטן או לפגוע בגנים שמרסנים את התקדמות המחלה. בנוסף, שגיאות מבניות יכולות לשנות את מבנה הגנים ואת הבקרה עליהם וליצור גנים חדשים.

השיטות המקוריות לגילוי של שגיאות בכרומוזומים השתמשו בצביעה ומיקרוסקופיה כדי לזהות בעין את השינויים. שימוש במערכים אליהם נדבקים מקטעי ד.נ.א אפשרו מדידת מספר עותקים באיכות גבוהה יותר. כיום, טכנולוגיות ריצוף עמוק מהוות את מקור המידע העיקרי לזיהוי שגיאות בסרטן. מקטעי ד.נ.א קצרים שנקראים מהגנום משמשים לשחזורו, והשוואה לגנום יחוס בריא מאפשרת זיהוי של שינויים.

## תמצית

בעיות רה-ארגון של הגנום עולות הן באבולוציה של מינים והן בחקר הסרטן. מודלים בסיסיים לרה-ארגון של הגנום מניחים שהגנום מכיל רק עותק אחד מכל גן, והשינויים היחידים שאפשריים הם מבניים, כלומר שינוי הסדר בין המקטעים הגנומיים. אולם, שינויים כמותיים כגון מחיקות והכפלות, המשנים את מספר העותקים מכל גן, נצפו באבולוציה של מינים ובמיוחד בהתפתחות של סרטן. בתיזה הזו אנו מתארים את מחקרנו בו פיתחנו מודלים לשינויים מבניים וכמותיים בסרטן. המודלים שונים זה מזה בהנחות שהם מניחים על מבנה הגנום וסוגי השינויים שהם מתירים בהתפתחותו. אנו נותנים אלגוריתמים יעילים ותוצאות קושי למודלים אלו, ומשתמשים בהם לניתוח של גנומים סרטניים. התיזה הזו מקדמת את המחקר של רה-ארגון גנומי בשני אופנים. המודלים שלנו מרשים אוסף נרחב יותר של פעולות מאשר מודלים אחרים ולכן הם יותר מציאותיים. בנוסף, הם מנסים לשחזר את סדרת הפעולות השלמה של שינויים מבניים וכמותיים באבולוציה של סרטן.

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

הפקולטה למדעים מדויקים ע״ש ריימונד ובברלי סאקלר
בית הספר למדעי המחשב ע״ש בלבטניק

# מודלים לשינויים מבניים וכמותיים בסרטן

חיבור לשם קבלת תואר ״דוקטור לפילוסופיה״

מאת **רון זעירא**

בהנחייתו של פרופ׳ **רון שמיר**

הוגש לסנאט של אוניברסיטת ת״א

אוגוסט 2018